DTIC FILE COPY

AD-A223 806

REPORT DOCUMENTATION PAGE	Form Approved OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE June 1990	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE A Path Planning and Obstacle Avoidance Hybrid System Using a Connectionist Network		5. FUNDING NUMBERS
6. AUTHOR(S) Christopher Emmet Schuster		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rice University, P.O. Box 1892, Houston, TX 77251		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Student Detachment, Troop Brigade, U. S. Army Soldier Support Center, Fort Benjamin Harrison, IN 46216-5820		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution is unlimited.		12b. DISTRIBUTION CODE A

DTIC ELECTE JUL 11 1990 S B D

13. ABSTRACT (Maximum 200 words)

Automated path planning and obstacle avoidance has been the subject of intensive research in recent times. Most efforts in the field of semiautonomous mobile-robotic navigation involve using Artificial Intelligence search algorithms on a structured environment to achieve either good or optimal paths. Other approaches, such as incorporating Artificial Neural Networks, have also been explored. By implementing a hybrid system using the parallel-processing features of connectionist networks and simple localized search techniques, good paths can be generated using only low-level environmental sensory data. This system can negotiate structured two- and three- dimensional grid environments, from a start position to a goal, while avoiding all obstacles. Major advantages of this method are that solution paths are good in a global sense and path planning can be accomplished in real time if the system is implemented in customized parallel-processing hardware. This system has been proven effective in solving two- and three-dimensional maze-type environments. (KE) (—

14. SUBJECT TERMS Path Planning Obstacle Avoidance Navigation Neural Connectionist Network		15. NUMBER OF PAGES 144
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

90 07 11 118

Standard Form 298 (Rev 2-89) Prescribed by ANSI Std Z39-18 298-102

RICE UNIVERSITY

A PATH PLANNING AND OBSTACLE AVOIDANCE HYBRID SYSTEM USING A CONNECTIONIST NETWORK

by

CHRISTOPHER EMMET SCHUSTER

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

APPROVED, THESIS COMMITTEE

[illegible]
John B. Cheatham Jr., Professor
of Mechanical Engineering
Director

John E Akin
John E. Akin, Professor and Chair
of Mechanical Engineering and
Materials Science Department

[illegible]
Angelo Miele, Professor
of Aerospace Sciences and
Mathematical Sciences

Houston, Texas

June, 1990

Abstract

A PATH PLANNING AND OBSTACLE AVOIDANCE
HYBRID SYSTEM USING A CONNECTIONIST NETWORK

Christopher Emmet Schuster

Automated path planning and obstacle avoidance has been the subject of intensive research in recent times. Most efforts in the field of semiautonomous mobile-robotic navigation involve using Artificial Intelligence search algorithms on a structured environment to achieve either good or optimal paths. Other approaches, such as incorporating Artificial Neural Networks, have also been explored. By implementing a hybrid system using the parallel-processing features of connectionist networks and simple localized search techniques, good paths can be generated using only low-level environmental sensory data. This system can negotiate structured two- and three-dimensional grid environments, from a start position to a goal, while avoiding all obstacles. Major advantages of this method are that solution paths are good in a global sense and path planning can be accomplished in real time if the system is implemented in customized parallel-processing hardware. This system has been proven effective in solving two- and three-dimensional maze-type environments.

Acknowledgements

I would like to express my sincere appreciation to Dr. John B. Cheatham, Jr. for his helpful advice and patience in directing my research. I am also deeply indebted to John Norwood, Peter Weiland, Sarmad Adnan, Jack Chen, and Dr. Y.H. Lin who spent many hours helping me refine concepts and debug program code. I want to give a special thanks to William T. Atkinson for being a good friend and for his countless hours spent working on the concepts for and building of the Maze Machine. (Wire wrapping is no fun alone!) Finally, I wish to thank my parents Paul and Gertraud, wife Kristine and son David, who always encouraged and supported me and without whom this research and thesis would not have been possible.

I would like to acknowledge the support obtained from NASA/ Johnson Space Center through the Research Institute for Computing and Information Systems (RICIS), Contract: 9-16, and NASA/ JSC Grant NAG-9-372.

I also wish to thank the U.S. Army for my support and the opportunity to attend Rice University.

Table of Contents

Tables

Figures

CHAPTER 1

Introduction

An important use for path planning and obstacle avoidance systems is the control of semiautonomous mobile robots. Robotic navigation involves controlling the movement of a vehicle from one location to another. While this problem seems trivial for humans, such is not the case for our electro-mechanical creations. The problem can be further complicated when travel requires movement through an unstructured, possibly changing, and complex three-dimensional environment. Currently, most systems in use force the human operator/ user to constantly control the robot's movement through a tele-operations link, using wire, radio, or fiber-optics as the information transfer medium. Automating the basic navigation ability of a robot would greatly simplify the human tele-operation requirements and allow for greater concentration on whatever task is to be accomplished when the robot reaches its goal location.

At Rice University alone, the Mechanical Engineering Robotics Group has been actively investigating alternative strategies for robotic sensing, navigation and control. [See Weiland (1989), Wu (1989), Norwood (1989), Cheatham (1987 & 1989), Adnan (1990), and Regalbuto (1988 & 1990).] Possible applications of this technology are:

Semiautonomous navigation control system for the future NASA Mars Rover exploration vehicle,

Household mobile robotic aids for the severely handicapped,

Automated roadmap routing system for use by police, fire, and ambulance emergency vehicle drivers, as well as tourists, cabbies, delivery service vehicle operators, etc,

Mobile robotic vehicle movement control system for travel between and through building(s) to: deliver parts, mail, sentry/ security patrol, carry people, etc.

The list above only hints at the possibilities for a reliable automated navigation system which, when combined with the proper sensor/ feedback apparatus and mobile base, can be used for a very wide variety of purposes. As an example, Figure 1 shows a layout for one floor of a small office building. One of the goals of this research is to develop an efficient path planning system which would allow a robot to travel throughout a similar multistory building environment to deliver mail, materials, etc.

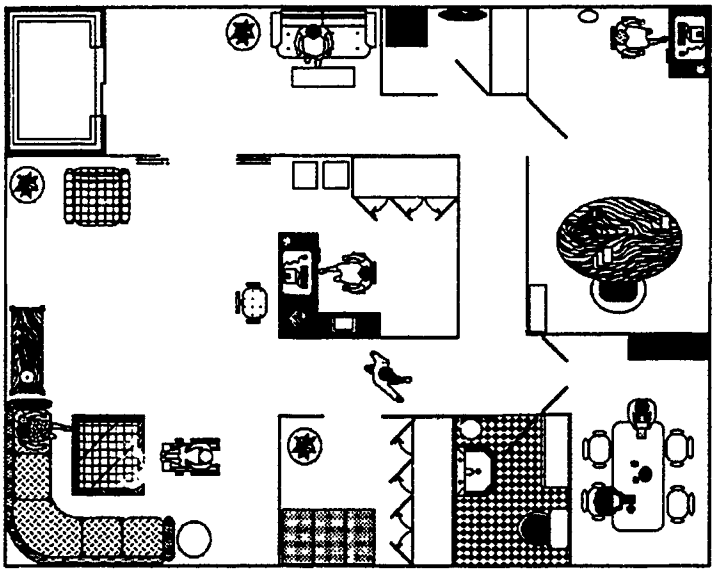


Figure 1

An Example Path Planning/ Obstacle Avoidance Environment

The assumptions made are: 1) there is a sensor/ feedback system in place which provides simple global environmental binary data input (i.e. represent space as a 3D 'grid' of obstacle vs. free space unit cubes), 2) the mobile robot is capable of 2D movement in eight directions (increments of 45 degrees), 3) the data grid scaling is suitable for considering the robot a point mass (or possible pre-processing of data to expand obstacles sufficiently to avoid collisions), and 4) other robot control routines are available to accurately update current position, handle opening/ movement through doors and elevators, and any other tasks desired.

Traditional methods for path planning and obstacle avoidance control use artificial intelligence search algorithms on sequential computers, however these algorithms normally do not provide real time control in a complex or changing environment and most are very susceptible to problems caused by noisy or incomplete environmental input data. For every special case, another routine has to be devised and programmed. One powerful computational alternative is a system featuring components roughly modeled on biological neural networks. These networks have advantages due to their highly interconnected/ parallel architectures.

The task investigated here is the use of a hybrid electronic connectionist network system for path planning and obstacle avoidance using only low-level environmental sensory input data. The system is called a hybrid due to its use of: 1) a combination binary input/ analog output resistive network which can be initialized, with sensory data, to represent a complex environment (called a connectionist network due to its highly interconnected nodal structure), 2) a second 'feed-forward' network which analyses the output of the connectionist network and provides local path move guidance, and 3) auxiliary digital control circuitry to handle the path finding procedure from start node to goal. By properly fixing the voltage outputs for start and goal points, along with modifying (disconnecting) connections to obstacles, the connectionist network can be caused to output useful information. The output of the connectionist network can be analyzed quickly by the second/ feed-forward local-move-finder network and additional circuitry to yield good path solutions for the given problem. The potential advantages of using this hybrid connectionist network system are speed, due to the massively parallel architecture; the ability to function reasonably with noisy and incomplete input; and efficient handling of changing environmental parameters. Neural-type networks have also been found to have an uncanny ability to survive minor damage/ loss of internal circuitry (much like the human brain, which functions even though neurons are constantly expiring).

This research was divided into two phases. First was the design and construction of a hardware implementation of a modest hybrid connectionist network system, here-on called the Maze Machine. The machine was built to prove and refine concepts and demonstrate feasibility for a much more ambitious VLSI implementation. The second phase of the research consisted of writing the software program AMAZ3D which simulates the Maze Machine's behavior on a sequential computer and allows for testing of much more complex navigation problems.

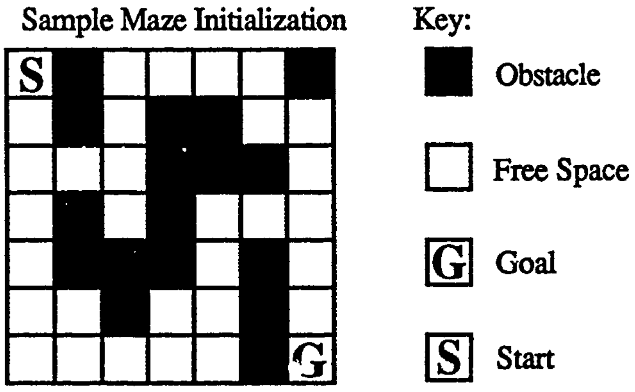


Figure 2

A Specific Maze Machine Problem Environment

The Maze Machine took the first step by solving simple two-dimensional 'maze' navigation problems. With the assistance of William T. Atkinson, a small scale hybrid electronic connectionist device was designed and built which can solve a subset of the desired navigation problem. The subset problem consists of movement through a two-dimensional environment where only four moves are allowed from a given point. Using Cartesian space, these directions are plus or minus X and plus or minus Y (later referred to as East, West, North, and South respectively). The two-dimensional environment is further limited to a seven-by-seven grid, or matrix, of possible locations. Each location, or 'node', will be externally designated as of four types: 1) obstacle, 2) free space, 3) goal, or 4) start. The problem environment is graphically shown in Figure 2. Note that the outer boundary around the maze is also treated as an 'obstacle' region, therefore movement is constrained to remain in the seven-by-seven maze.

Having a working version of the hybrid system not only demonstrated the validity of the concepts presented, but also provided the drive to conduct further research into possible construction of systems with much greater capability. Rather than immediately attempt the construction of custom analog VLSI chips to extend the capabilities of the Maze Machine, the second phase of the research consisted of writing a software program to simulate the operation of the hybrid system on a sequential computer. Thus the system's capabilities could be greatly expanded and many more tests could be run cheaply, while only sacrificing the speed advantage that would be gained by a true parallel processing network system.

The current version of the AMAZ3D program is capable of handling any size three-dimensional grid-type environment (limited only by the memory capacity of the computer being used). The allowable move directions from a given point/ node have been expanded to allow for eight directions in the horizontal plane (labeled North, South, East, West, North-East, North-West, South-East, and South-West) and two directions in the Vertical plane (labeled Up and Down). These added capabilities allow for the creation of robotic navigation environments which can reasonably simulate multistory building floorplans (where the robot takes elevators to reach different floors) or simple modelling of large scale outdoor terrain.

CHAPTER 2

Background

The navigation problem can be viewed as a search for a path (from current location to a goal) through an environment which contains obstacles. There are many different procedures that have been developed to solve these 'search' problems. Some procedures focus on finding feasible paths, while more complicated methods concentrate on finding optimal (i.e. shortest distance, least energy, etc.) paths. Implementation of these procedures is normally through the use of Artificial Intelligence (AI) software programs on sequential computers. In recent years, alternative methods, derived from the study of biological neural networks have been examined. These networks, although structurally very different, can provide results which closely resemble the good, and some times optimal, solution path AI methods.

Traditional Search Algorithms Approach

Basic Artificial Intelligence search procedures used to find feasible, and sometimes good, paths include Depth-first, Breadth-first, Hill climbing, Beam, and Best-first searches. [See Rich (1983), pp. 71-107, and Winston (1984), pp. 87-100.] These methods can be used when the length/ cost of the discovered path is not critical. The procedures follow specific algorithms to systematically search for a path from a start position to a goal position. Figure 3 is an example 'tree'-like roadmap through the space of possibilities for the 'maze'-type problem. As can be seen, from any given node there are four (for this example) children nodes/ possible moves (to its right). From any given child node there are again four follow-on moves that can be made. The basic search procedures listed above use different algorithms, but all result in the determination of a complete path from designated start to goal positions. For each method, if a solution exists, it will be found. However, there is no guarantee that it is the shortest/ optimal path. These procedures will also vary in the amount of time/ wasted steps taken before a feasible path is found.

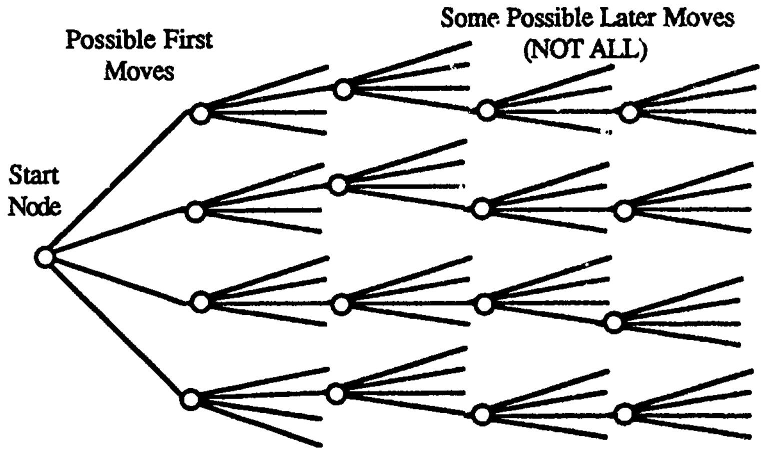


Figure 3

Diagram of Expansion of Potential Moves for Search Algorithms

To aid in understanding these search procedures, the Breadth-first algorithm will be explained in greater detail. The basic algorithm is shown in pseudo-code below:

```
To conduct a Breadth-first search:  [Winston (1984), pp. 95]
1.    Form a one-element queue consisting of the start node.
2.    Until the queue is empty or the goal has is reached,
      determine if the first element in the queue is the goal node.
      2a.    If the first element is the goal node, do nothing.
      2b.    If the first element is not the goal node, remove the
             first element from the queue and add the first element's
             children, if any, to the back of the queue.
3.    If the goal has been found, announce success; otherwise announce failure.
```

Breadth-first search looks for the goal node among all the nodes at a given level (equal number of steps from the start node) before using the 'children' of those nodes to push on. The procedure would then move on, level by level, until the goal is found, or no more moves are possible.

Procedures used to solve for optimal paths include Random-Exhaustive-Search (British Museum), Branch-and-Bound, Dynamic programming, and the A* search methods. [Winston (1984), pp. 101-113, Rich (1983), pp. 73-86] These procedures are utilized when it is important to minimize the length (or cost function) of the path. The Random-Exhaustive-Search approach finds all possible paths and then selects the best one; therefore, it is a very computationally expensive method. The A* procedure is designed to work efficiently, that is to say that it discovers an optimal path while expending minimum effort in finding the path. The A* search is much more efficient than the Random-Exhaustive-Search method and is really a combination of the two other mentioned optimal-path search procedures (Branch-and-Bound and Dynamic programming). The A* algorithm is shown in pseudo-code below:

```
To do A* search with lower-bound estimates: [Winston (1984), pp. 113]
1.    Form a queue of partial paths. Let the initial queue consist of the
      zero-length, zero-step path from the start node to nowhere.
2.    Until the queue is empty or the goal has is reached,
      determine if the first path in the queue reaches the goal node.
      2a.    If the first path reaches the goal node, do nothing.
      2b.    If the first path does not reach the goal node:
             2b1.   Remove the first path from the queue.
             2b2.   Form new paths from the removed path by extending
                    one step.
             2b3.   Add the new paths to the queue.
             2b4.   Sort the queue by the sum of the cost accumulated so far
                    and a lower-bound estimate of the remaining, with least-cost
                    paths in front.
             2b5.   If two or more paths reach a common node, delete all those
paths except for the one that reaches the common node with the minimum distance/ cost.
3.    If the goal has been found, announce success; otherwise announce failure.
```

A* pursues the most-likely shortest path by first checking all of the nodes that immediately follow the start node and creating a prioritized queue based on an estimated total distance to the goal node. Then it takes the (assumed) best partial path and creates new path extensions based on its new state. A* then resorts the queue of partial paths with least cost paths in front. (For extra efficiency, A* also removes redundant / inefficient partial paths which lead to the same node in the queue.) These checks continue as A* compares the lengths of the paths (i.e. the total known distance travelled plus the estimated distance remaining) and moves along the shortest path to the goal. Note: 1) A* will find the optimal path if the 'estimated' distance remaining to the goal is a lower bound on the actual distance. 2) A* wastes time when it checks potential paths that result in dead ends. 3) The performance of A* is dependent upon using reasonably accurate estimated distances between nodes. 4) The Random-Exhaustive-Search's disadvantage, as stated earlier, is that it is very computation/ time-consuming.

Summary of AI search procedures: [paraphrased from Winston (1984), pp. 131]

Depth-first and Breadth-first search are the simplest procedures. Both may be considerably less efficient relative to more 'informed'/ heuristic based procedures.

Hill climbing is a more informed procedure that explores 'tree' branches in the order of their heuristically guessed plausibility. Hill climbing shares a problem with its cousin, Depth-first search, in that a wrong decision early on can lead to useless wandering later. Hill climbing can also run into trouble if local maxima exist in the problem environment.

Beam search is a modification of Breadth-first search in which only the best nodes at any level are retained for further search. Beam search may fail to find legitimate paths.

Best-first searches push forward from the most promising open node yet encountered.

Branch-and-Bound search is a fundamental procedure for finding optimal paths. The basic idea is to extend the developing 'tree' from the end of the least costly partial path. Branch-and-Bound search is often improved through the use of estimates of distances remaining to the goal and by eliminating redundant paths to intermediate nodes, thereby becoming A*.

Several recent papers are included in the bibliography which concern topics which relate to path planning using variations of the traditional search methods. Regalbuto (1990) uses the A* algorithm as part of a control system for a mobile robot designed to help the severely handicapped. Mitchell (1988) reviews several methods for optimal path planning (including A*) combined with computational geometry terrain analysis techniques. Badreddin (1990) use a Best-first algorithm in combination with a geometrical and logical model of the environment and an associative memory for storing the paths. Tilove (1990) experimented with the Hill climbing search algorithm as part of a mobile robot local obstacle avoidance system based on the method of potential fields.

A Neural Network Approach

An interesting alternative to the traditional AI search techniques, all of which require large computational efforts on sequential computers, is the use of a customized artificial neural networks. The reference to neural networks comes from the study of biological neural networks such as the human brain/ nervous system.

Due to billions of highly interconnected neurons, the human brain is capable of solving complex problems, such as pattern recognition, very quickly. Many such tasks are still beyond the capabilities of our best algorithms even when implemented on the fastest supercomputers. The key difference is the use of a large number of simple processing elements (neurons) in parallel as opposed to a single complex central processing unit (CPU) handling information in a sequential manner. It should be made clear that no current artificial neural networks even slightly approach the complexities of the human brain, however significant advances in perception, cognition, and adaptation have been made by exploiting some of the features of the biological networks.

The basic building block of a biological neural network is the neuron. A neuron is a cell in the nervous system with the special characteristics of electro-chemical excitability.

Due to this excitability, the cell is able to conduct and process information. Figure 4 shows a sketch of a typical biological neuron, with key components labeled.

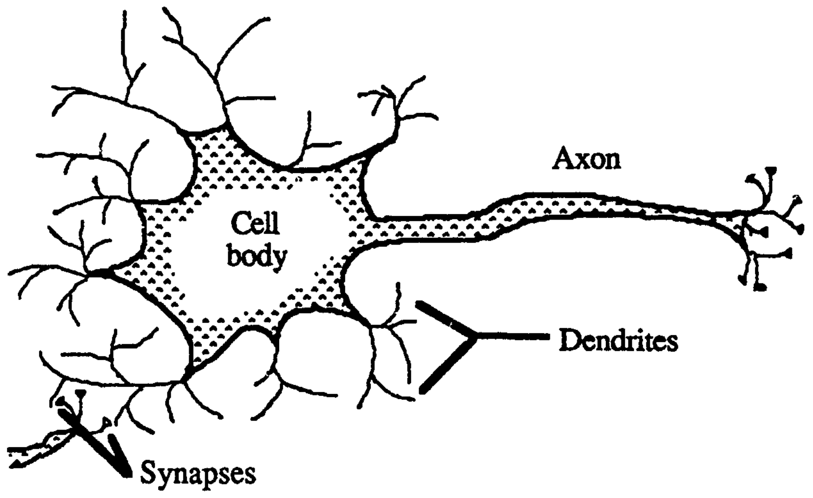


Figure 4

Biological Neuron

Main components of a neuron are: [See Wasserman (1989), Appdx. A, and Mead (1989), Chap. 4, for further details.]

> **Synapses:** junctions which form (usually) between the terminals of an axon and the dendrites of other neurons. They allow passage of information/ signals between cells.
>
> **Dendrites:** 'tree'-like extensions/ processes, which receive signals from other cells at junction/ connection points called synapses.
>
> **Cell body:** essentially averages the various signals received by the dendrites, thus determining the cells excitation level (if the signal average over a short time interval is sufficiently large, the cell 'fires').
>
> **Axon:** when the cell 'fires', a pulse is produced down its axon that is passed as signals to succeeding cells.

Note that hundreds of neuron 'types' have been identified, each with a distinctively shaped cell body and found to exhibit important functional specializations.

The information processing in the brain consists of a combination of chemical signals sent across the synapses and electrical signals within the neuron. It is the complex action of the cell membrane that creates the cell's ability to produce and transmit both kinds of signals. Note that the 'firing rate' of a neuron is determined by the cumulative effect of a large number of excitatory and inhibitory inputs, roughly averaged by the cell body over a short time interval. In this way, the neuron signal is 'pulse-rate' or frequency modulated.

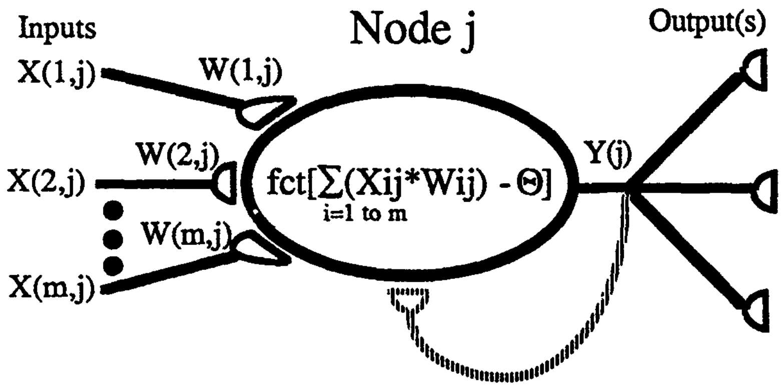
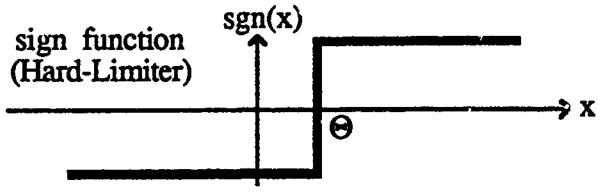


Figure 5

Artificial Neuron Model

Figure 5 shows a model of an artificial neuron. This model was designed to mimic the first-order characteristics of the biological neuron and is typical of the type currently being used in most neural network research. This sample neuron (Node j) has several inputs (Xij) with associated weights (Wij), a fixed threshold (Θ), and an output (Yj) (which may even reverberate back to Node j). Note that the function (fct) which is applied to the sum of the inputs multiplied by their weights and then modified by subtracting the threshold is normally a non-linearity function. Early/ simpler models often used the 'sign' function (hard-limiter) shown below:



For discrete time (and assuming the 'sign' non-linear function) the neuron model's output can be described mathematically as:

$$Y_j(t+\Delta t) = \text{sgn}\left[\sum_{i=1}^{m} w_{ij} X_i(t) - \theta_j\right]$$

Other non-linearity functions, such as and the sigmoid/ logistics function (which allows a range between 0 and 1), are used more often now due to improved modelling characteristics and continuous differentiability.

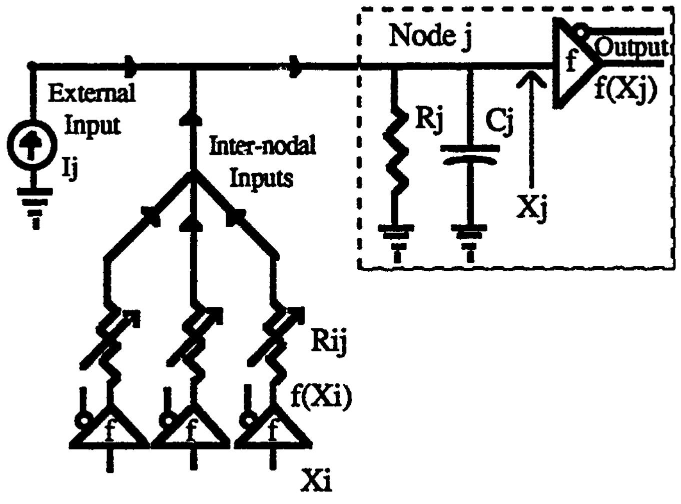


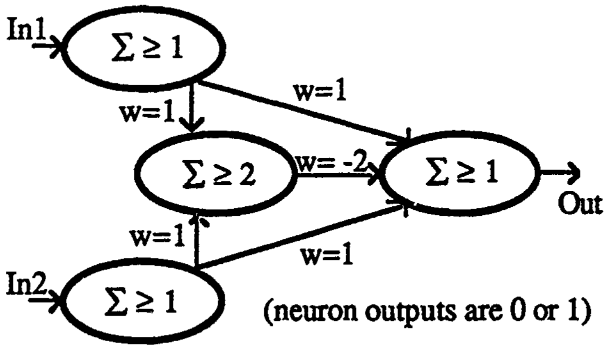
Figure 6

An Electrical Circuit Neuron Model

An electrical circuit model for a basic (additive subclass) neuron is shown in Figure 6. It can be mathematically described by the following equation: [ref: Ögmen (1989)]

$$C_j \frac{dX_j}{dt} = -\frac{1}{R_j} X_j + \sum_{i \neq j} \left[f(X_i) - X_j \right] \frac{1}{R_{ij}} + I_j$$

So far the neural network background review has covered a single neuron. Although understanding a neuron's function is very important, the 'power' of neural networks comes from their parallel/ highly-interconnected structure, memory, and learning/ adapting capabilities. While the interconnected structure is visually obvious, the memory of a network is hidden in the values of the connection weights and thresholds. No individual neuron is used to store a complete 'single-memory', rather the network as a whole responds to its input by producing a global (across the network) response based on all stored connection weights and threshold values. Learning by the network involves modifying these weights and thresholds to adapt the response to given input(s). An example of a simple binary network which has been trained/ taught to solve the classic EXclusive-OR logic function is shown in Figure 7. The connection weights and neuron thresholds have already been set appropriately, based on the input versus the desired output set. (Here the nodal non-linearity function consist of outputing a 0 or 1, depending on the False or True result of the threshold inequality, respectively.)



XOR Table

Input In1	Input In2	Output Out
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7

Binary Neural Network Taught to Perform EXclusive-OR Function

Several recent papers are included in the bibliography which concern topics relevant to using artificial neural networks for navigation. Norwood (1989) uses an neural-type network to create potential fields as part of a robotic path planning/ obstacle avoidance system. Hopfield (1985) uses recurrent networks to solve the classic 'Traveling Salesman Problem' (results are not guaranteed to be optimal, yet 'good' solutions are reached rapidly for even very complex cases). Badreddin (1990) uses an associative memory (often implemented as neural networks) to store available paths in connection with a geometrical and logical environmental modelling scheme. Hutchinson (1988) uses a combination analog/ binary resistive network for computing optical flow as part of a biological early vision model.

CHAPTER 3

Hybrid Network System for Navigation

As stated earlier, a goal of this research project is to design and build an electronic connectionist network system which will solve a specific two-dimensional maze-type navigation problem. To solve this problem a hybrid system was designed which contains two separate and very different network structures brought together with a supervisory/ control system. The first network processes the environmental sensory input using a binary input/ analog output resistive 'connectionist' network. The output of the connectionist network is repeatedly analyzed by the second network, a small feed-forward analog input/ binary output network, which determines the local path moves.

Sensory Analysis Connectionist Network

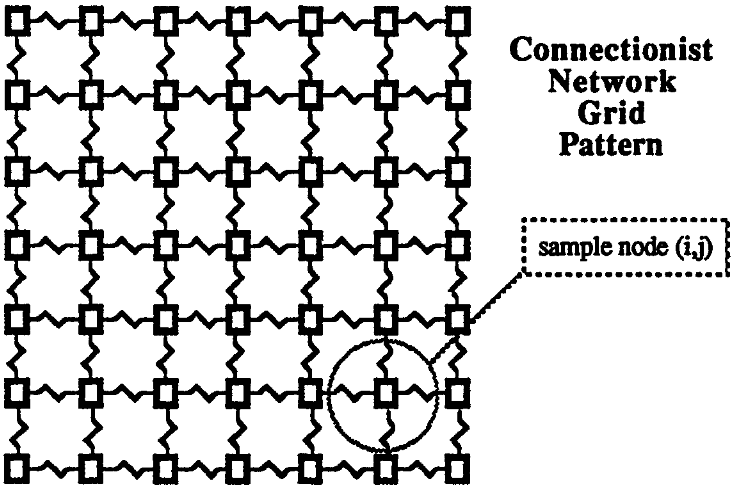


Figure 8

Connectionist Network Structure

The first network consists of a seven-by-seven grid of simple processing 'nodes' as shown in Figure 8. Each node is connected to its four nearest neighbors and receives external input setting it as one of the four types mentioned earlier (obstacle, free space, goal, or start). A sample neighborhood for the sample Node i,j is shown in Figure 9. The inputs to this network designate each node as either obstacle, free space, goal, or start.

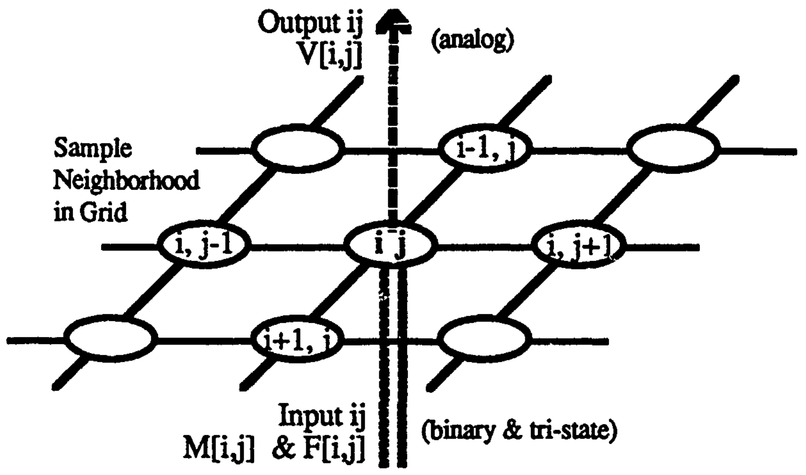


Figure 9

Connectionist Network Sample Node

A computationally convenient choice for the input is by way of two seven-by-seven matrices. Matrix **M** (Maze Mask Matrix) contains elements which are set to 0 if the node is an obstacle and a 1 if it is a free space, goal, or start. The second matrix, **F** (Forced Nodes Matrix) contains tri-state elements. A goal node is represented by a 1, the start and obstacle nodes are represented by a -1, and free space nodes are represented by a 0. The network's output is represented by an analog seven-by-seven matrix **V**. Elements of **V** range anywhere between -1 and 1. **V** represents an 'energy potential' map of the maze. The values of forced nodes are known 'up-front', however a free space node's output is the average of the outputs of its four nearest neighbors which are non-obstacle or non-boundary region. Optionally, matrix VSE, which is equivalent to V except that the range of its elements are GND (0) to VCC, provides network output using the same scale as the electronic implementation. Based on the above input definitions we can analyze the output for any given Node i,j (at row i, column j). The possible outputs are:

Goal

$V_{ij} = 1$ (VCC for electronic implementation)

Start / Obstacle

$V_{ij} = -1$ (GND, 0 for electronic implementation)

Free Space

$$V_{ij} = \left[\frac{\sum_{\text{neighbors}} V_{kl} M_{kl}}{\sum_{\text{neighbors}} M_{kl}} \right]$$

where **neighbors (k,l)** are the four nearest nodes:

(i-1, j)	up
(i, j+1)	right
(i+1, j)	down
(i, j-1)	left

Note: If a free space node is surrounded by four obstacle nodes, this equation would result in division by zero. This can be corrected with a conditional test where if $\sum M_{kl} = 0$, then arbitrarily set $V_{ij} = -1$. This does not cause a problem for the electronic implementation, since such a node (which would have a 'floating' value) would never be reached by the solution path.

A '**General Node Output Equation**' can be represented by: (see **Note** above)

$$V_{ij} = \left[\frac{\sum_{\text{neighbors}} V_{kl} M_{kl}}{\sum_{\text{neighbors}} M_{kl}} \right] \left[1 - (F_{ij})^2 \right] + F_{ij}$$

An examination of the general output equation makes it clear that computing the output for this network consists of solving forty-nine simultaneous linear equations of up to forty-seven unknowns (assuming the case where there is one goal, one start node and no obstacles). The solution will take some time on a digital computer, however it is almost instantaneous on the parallel processing network. For the optional output matrix **VSE**, the General Node Output Equation (to simulate the electronic implementation) can be described by: (see **Note** above)

$$\mathbf{VSE_{ij}} = \left(\left[\frac{\sum_{\text{neighbors}} \mathbf{V_{kl} M_{kl}}}{\sum_{\text{neighbors}} \mathbf{M_{kl}}} \right] \left[1 - (\mathbf{F_{ij}})^2 \right] + \mathbf{F_{ij}} \right) \frac{\mathbf{Vcc}}{2} + \frac{\mathbf{Vcc}}{2}$$

Vcc is the source/ supply voltage for the electronic implementation.

Local-Move-Finder Network

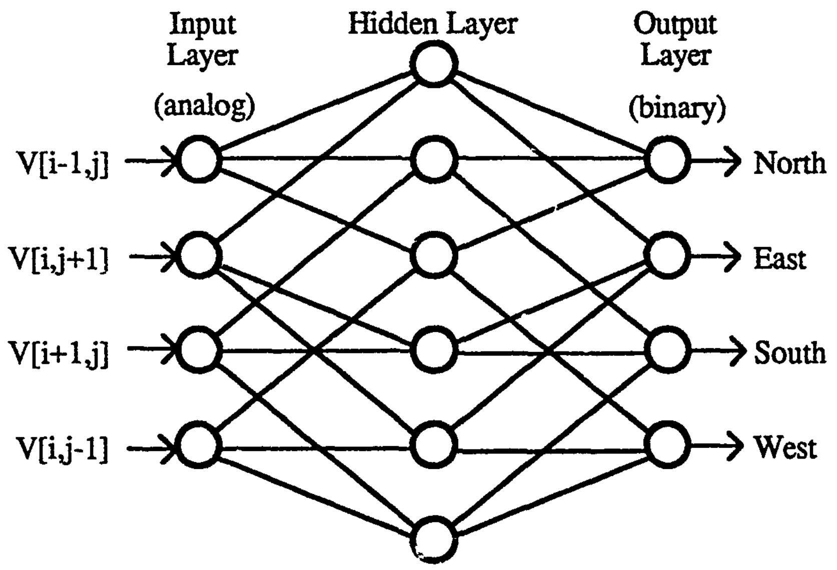


Figure 10

Local-Move-Finder Network Structure

The second network consists of a three-layer feed-forward system with fourteen nodes (4 input, 6 hidden, and 4 output). In the input layer, four nodes receive analog inputs from the previous network through a control circuit which selects the four neighbor node outputs for any 'given' current location. The network processes the input through six hidden nodes and the four output nodes provide the information as to the 'best' move direction from the current position (a 'winner-take-all' binary output signal). Figure 10 shows the structure for this network.

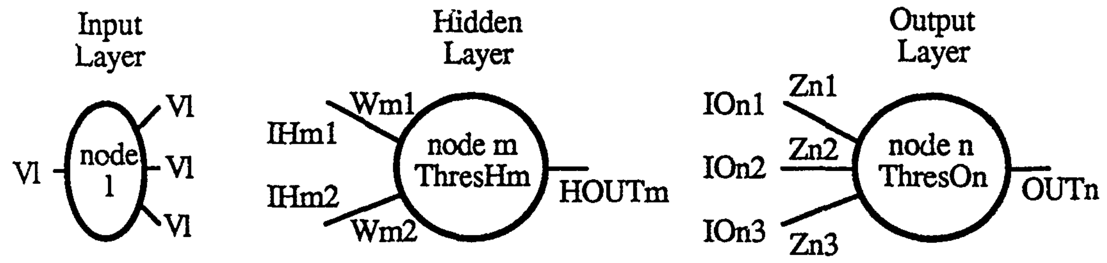


Figure 11

Local-Move-Finder Network Sample Nodes

Figure 11 shows a sample node from each layer of this network. General output equations for the nodes of each layer can now be derived. The Input Layer (node l) is purely a distribution layer, taking its analog input signal and channeling it to three destination nodes of the second (hidden) layer. Each Hidden Layer Node (example node m) receives two inputs: IHm1 and IHm2. Each input is multiplied by its appropriate weight ,Wm1 or Wm2, and these values are then summed. We subtract the threshold value, ThresHm, from the input sum and then apply the sign (Non-Linearity) function to this value. Thus if the value is negative, the output from the Hidden Layer would be HOUTm = -1; if the value is positive, the output would be HOUTm = +1. For this particular application, set Wm1 = +1, Wm2 = -1, and ThresHm = 0. This results in a simple comparator node, where the output is 'high' if the first input value is greater than the second and 'low' if the first input is less than the second. Thus the output from a Hidden Node can be described by:

$$\mathbf{HOUTm} = \mathrm{SGN}[\sum_{i=1,2}(\mathrm{IHmi})(\mathrm{Wmi}) - \mathrm{ThresHm}] = \mathrm{SGN}[\mathrm{IHm1} - \mathrm{IHm2}]$$

$$\text{where ThresHm} = 0, \quad \mathrm{Wm1} = +1, \quad \mathrm{Wm2} = -1$$

In the Output Layer, each node (example node n) receives three inputs: OHn1, OHn2 and OHn3. Each input is multiplied by its appropriate weight: Zn1, Zn2 or Zn3. These values are then summed. Next, the threshold value, ThresOn, is subtracted from the input sum and finally the sign (Non-Linearity) function is applied to this value. Thus if the value is negative, the output from the Output Layer would be OUTn = -1; if the value is positive, the output would be OUTn = +1. For this particular network, set ThresOn = 2.5 and the three weights appropriately to +/- 1 to match the three-input AND gates of Module 4 in Appendix A (-1 for AND inputs with inverters). Thus, the Output Layer's nodes output become:

$$\mathbf{OUTn} = \mathrm{SGN}[\sum_{i=1,2,3}(\mathrm{IOni})(\mathrm{Zni}) - \mathrm{ThresOn}]$$

$$\text{where ThresOn} = 2.5, \quad \mathrm{Zn1} = +/- 1, \quad \mathrm{Zn2} = +/- 1, \quad \mathrm{Zn3} = +/- 1$$

And individually:

$$\mathbf{OUT_1} = \mathrm{SGN}[\mathrm{IO}_{11} + \mathrm{IO}_{12} + \mathrm{IO}_{13} - 2.5]$$

$$\mathbf{OUT_2} = \mathrm{SGN}[-\mathrm{IO}_{21} + \mathrm{IO}_{22} + \mathrm{IO}_{23} - 2.5]$$

$$\mathbf{OUT_3} = \mathrm{SGN}[-\mathrm{IO}_{31} - \mathrm{IO}_{32} + \mathrm{IO}_{33} - 2.5]$$

$$\mathbf{OUT_4} = \mathrm{SGN}[-\mathrm{IO}_{41} - \mathrm{IO}_{42} - \mathrm{IO}_{43} - 2.5]$$

CHAPTER 4

Hardware Implementation of Hybrid System

Chapter 3 covered the theory behind the two network architectures proposed for use in navigation control. This chapter covers the actual implementation of the hybrid system using electronic hardware. The resulting device was named the 'Maze Machine'. It consists not only of the two networks, but also the accessory electronic interface circuitry (required for a simple stand-alone system) needed to provide the inputs, multiplexing, control, and path display of solutions .

The Maze Machine

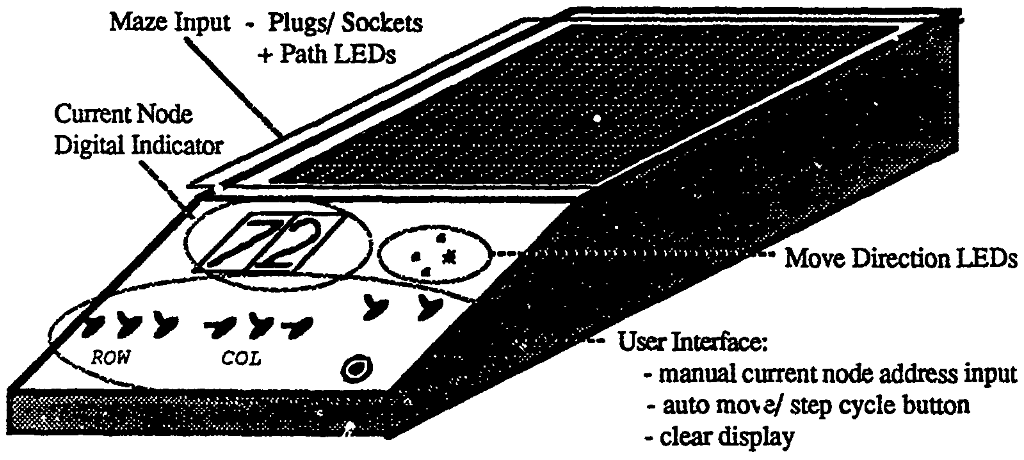


Figure 12

Sketch of Maze Machine

The design and construction of the hardware system resulted in the device shown in Figure 12. It is slightly smaller than a 'bread box' and is a self contained unit capable of solving the seven-by-seven grid 'maze' problems explained earlier. The device was constructed using six 'modules' that, when properly inter-connected, make up the Maze Machine. Figure 13 shows a block diagram of the layout and information exchange characteristics of the system modules.

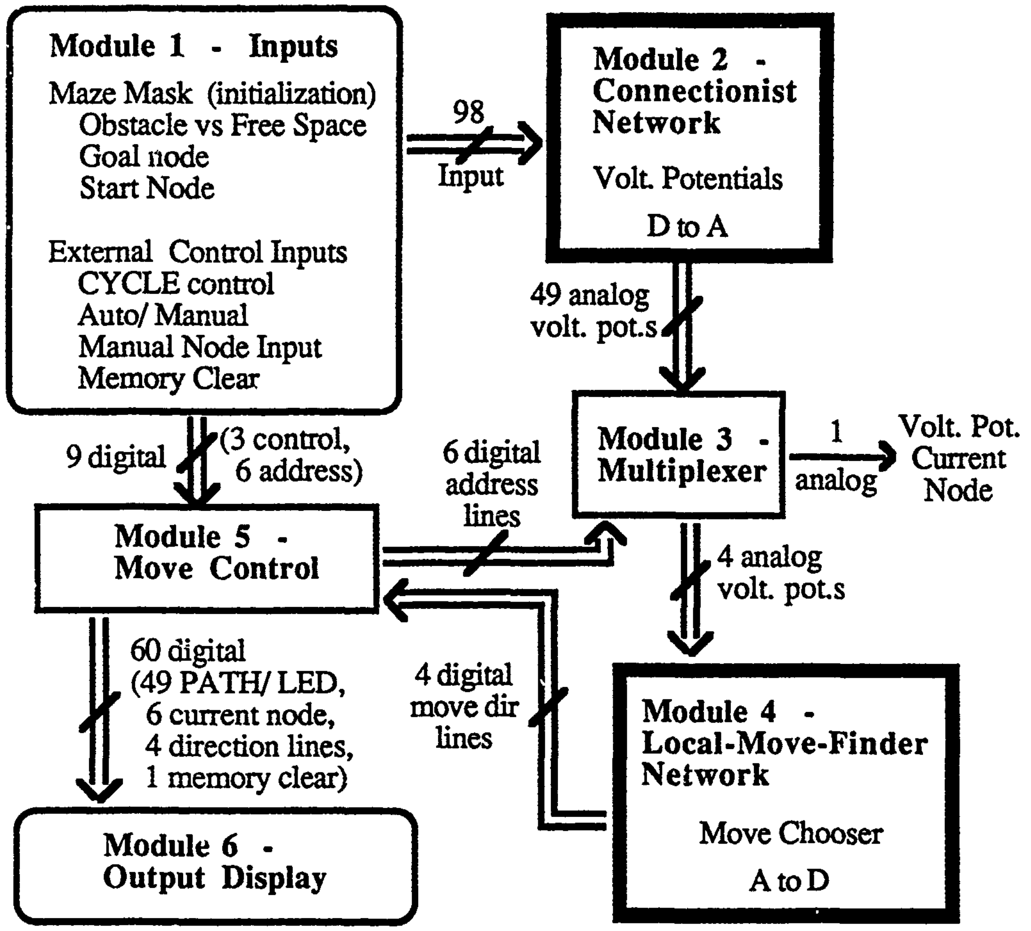


Figure 13

Block Diagram of Electronic Hybrid System

For detailed graphical views of the individual modules see Appendix A, Maze Machine Electronic Module Diagrams. For module/ IC chip wiring/ connection information see Appendix B, Maze Machine Module Wiring Tables. For a parts inventory and assembly notes see Appendix C, Maze Machine Tools/ Parts Required. What follows is a module by module description of the design and operation of the Maze Machine:

Module 1 - Input) This is the input block (hand-placed electrically-wired plugs which fit into sockets of Module 2). Four kinds of plugs are used to represent the four possible conditions of a node: obstacle, free space, goal or start. The particulars for each of the four node types are: 1) for an obstacle node (and also the outer boundary), force the node voltage output to GND and disconnect its resistor connection links to its neighbors, 2) for a free space node, allow the node voltage output to 'settle' on the average value of the neighbor nodes influencing it (while also allowing its output to influence its neighbors, i.e. recurrent behavior), 3) for the goal node, force the node voltage output to VCC (source/ supply voltage) and allow it to influence its neighboring nodes, and 4) for the start node, force the node voltage output to GND and allow it to influence its neighboring nodes. (user controlled binary data) (see Appendix A, Figure A1)

Module 2 - Connectionist Network) This network consists of 49 wired sockets interconnected by 100 kΩ resistors. These nodes receive binary inputs through the input plugs and the nodes provide appropriate output 'voltage potentials' (analog) derived as a function of the various external inputs and the interconnections between neighboring nodes. Note that no pins within a node socket are connected to each other until one of the four plugs of Module 1 is inserted. (binary to analog) (see Appendix A, Figure A2 & A3)

Module 3 - Multiplexer) This module is a multiplexing circuit which receives the 49 analog 'voltage potentials' from Module 2 and the current binary coded decimal (BCD) location address (representing the robot's position in the maze) from the Move Control Module. The module's output consists of five lines. The first four lines carry the 'voltage potentials' of the four nearest neighbors of the current node to the Local-Move-Finder Network, Module 4. The fifth line provides the current node's 'voltage potential' as a separate (external) output for test verification and review. (analog+binary to analog) (see Appendix A, Figure A4 & A5)

Module 4 - Local-Move-Finder Network) This network takes the four analog 'voltage potentials' from the Multiplexer, Module 3, and provides as its output four binary lines which tell the Move Control Module and the user which direction, of the four possibilities, shows the greatest voltage increase and is thereby the 'best' move (locally). As with the network design, the comparison is internally done in parallel for speed advantages. (analog to binary) (see Appendix A, Figure A6)

Module 5 - Move Control) This module provides the cycle control which steps the machine through the navigation solution path, one step/ move at a time, by providing the appropriate current location address to the Multiplexer Module and updating the current location, as required, based on the Local-Move-Finder Network Module's output. This module can be provided with manual/ user controlled external inputs for setting the current address to any desired value for running special tests (see Appendix D, Procedure for Running Maze Machine Auto-Path Test). This circuit also sends the current address to the Path Output Display Module. (binary) (see Appendix A, Figure A7)

Module 6 - Path Output Display) This module contains multiplexing circuitry and 49 RS flip-flops, one per node. It controls an array of 49 LEDs which display the solution path (and the intermediate moves as the path is being generated). (binary) (see Appendix A, Figure A8)

Note: The plugs and sockets of Modules 1 and 2 are an awkward system used for its relative simplicity and low cost. Figure A9, Appendix A, shows an alternative which replaces the plug and socket nodes with a set of wired IC chips (3 off-the-shelf CMOS chips required per node). These nodes would be controlled through flip-flop memory which could be quickly initialized with test input data using a serial computer interface.

Maze Machine Results

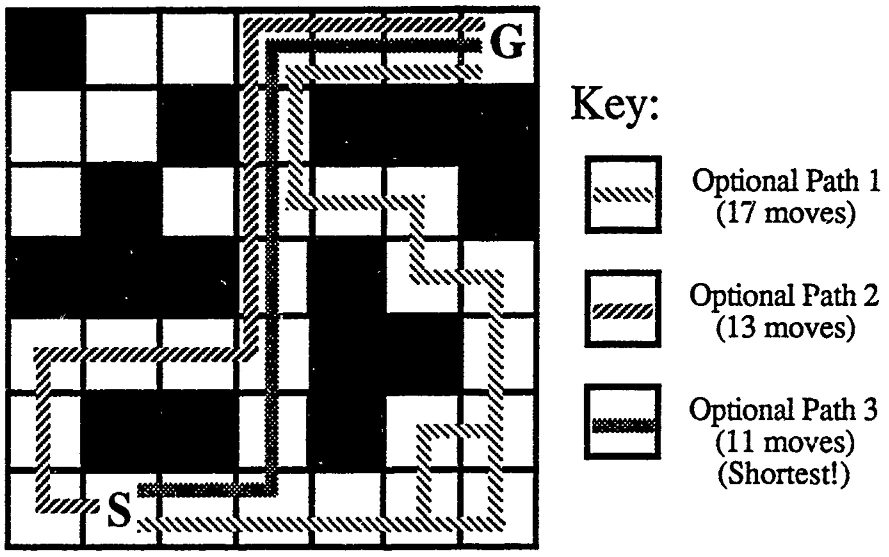

G
S
Key:
Optional Path 1 (17 moves)
Optional Path 2 (13 moves)
Optional Path 3 (11 moves) (Shortest!)

Figure 14

Sample Maze Machine Test Problem Analysis

The Maze Machine was tested using a variety of sample 'mazes'/ path planning and obstacle avoidance problems. The machine consistently provided good (often optimal) solutions based on the environment and assumptions imposed. Figure 14 shows a typical test 'maze', along with all possible solution paths.

	1	2	3	4	5	6	7
1	0.00	6.86	6.86	**6.86**	**8.03**	**9.29**	**10.53**
2	6.72	6.76	0.00	**5.60**	0.00	0.00	0.00
3	6.62	0.00	4.44	**4.45**	4.07	3.80	0.00
4	0.00	0.00	0.00	**3.47**	0.00	3.51	3.20
5	1.33	1.72	2.13	**2.56**	0.00	0.00	2.91
6	0.87	0.00	0.00	**2.06**	0.00	2.40	2.70
7	0.44	**Start**	**0.80**	**1.60**	1.83	2.12	2.09

Table 1

Voltage Potential Table for Maze Machine Sample Test Problem

Table 1 shows the nodal voltage potentials for this problem. The solution path is highlighted in bold print. (Reference Appendix E, Maze Machine Test Data and Results, Test 4 -Table E6.) The path selected by the Maze Machine for this particular test was Option Path 3 (which was shortest/ optimal).

In the tests ran, the environment defined usually contained more than one feasible path (from start to goal). After setting the environment (by defining start, goal, free space, and obstacle nodes), voltage readings were taken for each node. As expected: 1) an obstacle had a voltage output of GND/ 0 volts (note that the boundaries were treated as obstacles), 2) the start point also had an output of GND/ 0 volts (it differed from obstacles in that the start point outputs its 'forced' voltage to its neighbors), 3) the goal was set/ 'forced' to source voltage, VCC/ approximately 12 volts, and 4) the free spaces had voltage readings ranging from GND to VCC volts. Next, the path was determined step-by-step, beginning at the start node. For each move, the voltage difference between the 'current' node and its four neighbors (i.e. north, east, south, west) was analyzed. The best (local) move was selected as being in the direction of the greatest voltage increase. By following the path of greatest local voltage increase, the path that the Maze Machine would select could be determined.

After sampling the voltage outputs, each actual test was run on the Maze Machine. As stated earlier, in each case, the machine successfully navigated the maze and avoided obstacles while selecting a good (best, from a local voltage increase basis) path. The path selected was indeed the same path that was found by following the path of greatest local voltage increase from the recorded voltage potential table. This electronic hybrid network system successfully solved the two-dimensional navigation problems that were presented. See Appendix E for the nodal voltage potentials and the machine's results for several sample mazes/ tests.

Chapter 5

Software Simulation of Hybrid System

The design and construction of the Maze Machine was the first phase of this research project. This device successfully demonstrated the real-time navigation control capability of the described hybrid network system. The plan now called for: 1) going beyond the seven-by-seven grid limitation, 2) allowing an option of eight possible horizontal moves from a given node (thus allowing the robot diagonal travel), and 3) allowing option of vertical (3D) movement (such as travel up and down a building's elevator). To build a reasonably sized system in hardware would likely require extensive use of custom analog VLSI technology. Rather than expend this large effort/ cost, the second phase of research consisted of writing a software program which could simulate the output characteristics of the parallel-architecture Maze Machine, while also permitting the extended capabilities mentioned above.

It should be made clear that this chapter describes a *simulation* for the desired system. A main advantage mentioned earlier, that of real-time control, is lost due to the serial/ iterative processing required for such a software based system. Note that the Maze Machine's connectionist network almost instantly 'settles' on stable nodal voltage potential outputs (no matter what the size of the grid environment, 2D or 3D), while the software program must process/ compute a large number of simultaneous linear equations (i.e. 49 equations of up to 47 unknowns for the 7-by-7 grid problems) in an iterative fashion. Thus the software solution is very time consuming and its efficiency is highly effected by the problem complexity/ grid size being analyzed, the memory capacity, and speed of the computer/ CPU being used.

The AMAZ3D Program

The result of the software simulation effort is a program named AMAZ3D. The source code was written in FORTRAN77 and has been successfully compiled and run on various computer systems (Macintosh PC, SUN workstation, and UNIX based network). The main program AMAZ3D.f, along with its subroutines, comprises the complete sequential computer simulation for the hybrid connectionist network system used for path planning/ obstacle avoidance. Some of the program's organization and operation show similarities to the traditional search algorithms mentioned in Chapter 2, Background (i.e. the Local-Move-Finder Network has been replaced by a Best-first search routine). The outline for AMAZ3D is shown in pseudo-code below:

```
To find a good path using AMAZ3D:
1.    Enter program preferences.
2.    Input problem parameters, environmental data, start and goal positions.
      2a.   Allow for modification of parameters and problem inputs.
3.    Initialize nodal array. Set: free space = GND (initially),
                                   obstacles = GND,
                                       start = GND, &
                                       goal = VCC.
4.    Calculate values for nodal voltage potentials by 'setting'
      free node Vouts to average of non-obstacle neighbors.
      4a.   Until # of iterations or accuracy is reached, return to 4.
5.    Output # of iterations and error estimate.
6.    Initiate PATH with current node position = start position.
7.    Calculate path:
      7a.   If current position = goal position, go to 8.
      7b.   Determine move direction (DIR),
            based on 'best' (local) move from current position.
            7b1.  If no volt increase move exists, announce failure, go to 9.
      7c.   Update PATH with new current position, based on old + DIR.
      7d.   Update path length info, return to 7.
8.    Announce success, output PATH/ Problem Solution.
9.    Return to 1. or exit program.
```

The actual AMAZ3D program reads in three data files:

1) **amaz3d.prf** [A preferences file for: a) describing the screen output device number; 6 for IBM PCs/ SUN workstations and 9 for the Macintosh PC and b) FILEDF, the default file-name for the assumed input parameter file.],

2) **FILEDF or file-name entered by user** [A parameter file which provides key information to the program. All parameters but the sensory data file array dimensions and FILEM, the initial node declarations file, can be modified later by the user within the AMAZ3D program.],

3) FILEM [Sensory data input file, name provided by above mentioned parameter file, which provides AMAZ3D program with initial environmental sensory data for the nodal array which was dimensioned through FILEDF entries.].

Note: In the software program it is convenient to enter the free space versus obstacle environmental-data as an array, the start and goal node positions as vectors, and several other program preferences as appropriate variables.

The program accomplishes its simulation of the parallel processing connectionist network by starting out with an initial unstable network output state. (Goal node set to VCC, all other nodes initially set to GND. Note that obstacle nodes are disconnected/ isolated from their neighbors.) Next AMAZ3D iteratively updates/ refines the network towards a stable output array state. One iteration consists of resetting each free node's next state to the average value of its 'connected' neighbor nodes' current states. In this way, after numerous iterations, the output array state of the nodal network will approach a stable value (with free node values between GND and VCC). The iteration stop/ cut off can be set to a maximum number of iterations or to a maximum allowable single iteration nodal change value.

After the network system has stabilized (to an acceptable degree), other subroutines (described individually in Appendix F) are used to determine the actual path by using a step-by-step examination of the current position node's nearest neighbors and selecting moves which follow the path of greatest local nodal value increase.

Since this program is only a simulation of the desired parallel processes of the Maze Machine, it has the unique feature of being able to provide the number of steps for (globally optimal) minimum-distance solutions to the path planning problems presented. This is accomplished by adding a step to the iterating process which checks to see if any of the start node's nearest neighbors have been disturbed (i.e. if any of their output values change from their initial GND value). If a disturbance is detected, the program alerts the user with the current number of iterations and presents the opportunity to stop the iterative process and move directly to the localized path-finding subroutines. The user-alert occurs at the minimum distance number of steps due to the fixed unit grid structure of the connectionist network. Since all inter-node connections are assumed to be of equal distance, the 'voltage' disturbance radiating out from the goal will travel an identical distance in all allowable directions from the goal for any given number of iterations. Therefore, the shortest distance between goal and start will provide the earliest disturbance to the start node's neighbors. If the iteration process is now stopped, the network will not have a chance to 'settle' but will still provide a good path solution. (Note that the actual paths are still found using the local optimizer routine which may provide non-optimal global solutions due to forks in the potential paths.) This extra check step in the iteration process can be described as a Breadth-first search technique (refer back to Chapter 2, p. 7 for pseudo-code) which has been conditioned to allow only equal length moves, thereby making the first path length detected also the shortest path by definition.

Finally, the program provides subroutines for displaying the path information in two formats: 1) a step-by-step move list, which provides each path nodes' grid location along with the direction to move and 2) a printout of the entire nodal environment with the path steps highlighted by numbers counting up from 1 to 9, then a 0, and starting again at 1 (this count cycle is used to minimize the size of the output printout, while still keeping critical information about move direction). (See Appendix F for AMAZ3D.f source code listing, and Appendix G for sample problem outputs.)

Note that to keep the program memory requirements reasonable for a microcomputer, the maximum sized sensory data input file has been limited to a three dimensional array of 80 rows by 80 columns by 8 layers of height (which requires approximately 1 Mbyte RAM). These values are arbitrary, of course, and can be changed in the variable declaration statements of the program source code before compilation; the only limitation is memory availability. Also note that two types of obstacles have been allowed for: 1) 'normal' obstacles are set to GND and are isolated from their neighbors in the network (i.e. do not influence the neighbors' nodal value outputs) and 2) 'connected' obstacles which have the same characteristics as the start node (i.e. set to GND but left connected to the network so that their influence is felt by the neighboring free nodes). This second type of obstacle node allows the program to approximate the potential fields approach used by Norwood (1989) in his Master's Thesis on Robotic Path Planning and Obstacle Avoidance: A Neural Network Approach. Paths created using the connected obstacles show the tendency of trying to 'avoid' the obstacles rather than 'side-swipe' them to minimize distance travelled.

AMAZ3D Results

The AMAZ3D program was tested using a variety of sample path planning/ obstacle avoidance problems. The software simulation system consistently provided good (often optional) solutions based on the environment and assumptions imposed.

	1	2	3	4	5	6	7
1	0.00	6.82	6.82	**6.83**	**8.06**	**9.30**	**10.53**
2	6.82	6.82	0.00	**5.60**	0.00	0.00	0.00
3	6.82	0.00	4.37	**4.37**	4.05	3.74	0.00
4	0.00	0.00	0.00	**3.46**	0.00	3.43	3.11
5	1.27	1 70	2.12	**2.55**	0.00	0.00	2.80
6	0.85	0.00	0.00	**2.06**	0.00	2.34	2.49
7	0.42	**Start**	**0.79**	**1.58**	1.88	2.19	2.34

Table 2

Voltage Potential Table for AMAZ3D Sample Test Problem

Table 2 shows the program's final nodal voltage potentials for the same test maze run on the Maze Machine and shown in Figure 14 of Chapter 4. (Also see Appendix E, Test 4 - Table E6 for complete Maze Machine results, and Appendix G, maz.out, for complete AMAZ3D printout.) The program was allowed to iterate through 371 cycles, until the maximum individual nodal change per iteration was less than 0.001.
Note that the solution path given AMAZ3D was the same as the Maze Machine. Table 3 shows the comparative differences between the nodal voltage potentials of the Maze Machine (see Chapter 4, Table 1) and AMAZ3D for the sample maze of Figure 4 (the numbers in the table represent Vout (AMAZ3D) - Vout (Maze Machine)).

	1	2	3	4	5	6	7
1	0.00	-0.04	-0.04	-0.03	0.03	0.01	0.00
2	0.10	0.06	0.00	0.00	0.00	0.00	0.00
3	0.20	0.00	-0.07	-0.08	-0.02	-0.06	0.00
4	0.00	0.00	0.00	-0.01	0.00	-0.08	-0.09
5	-0.06	-0.02	-0.01	-0.01	0.00	0.00	-0.11
6	-0.02	0.00	0.00	0.00	0.00	-0.06	-0.21
7	-0.02	0.00	-0.01	-0.02	0.05	0.07	0.25

Table 3

Voltage Potential Difference Table, Maze Machine vs. AMAZ3D

Note that node (7,7) has the largest error (approximately 12% difference), however, in general the two tables match fairly closely. As stated before, the differences can be explained by component tolerances in the Maze Machine (i.e. resistors), as well as some error in the AMAZ3D results due to iteration cut-off before absolute stability/ 'settling'.

Figure 15 shows another seven-by-seven maze example (again with only perpendicular moves are allowed). This problem is a sample of a case where the system produces a non-optimal solution path (see Appendix G, mazno.out). Table 4 shows the AMAZ3D voltage potential table for this problem.

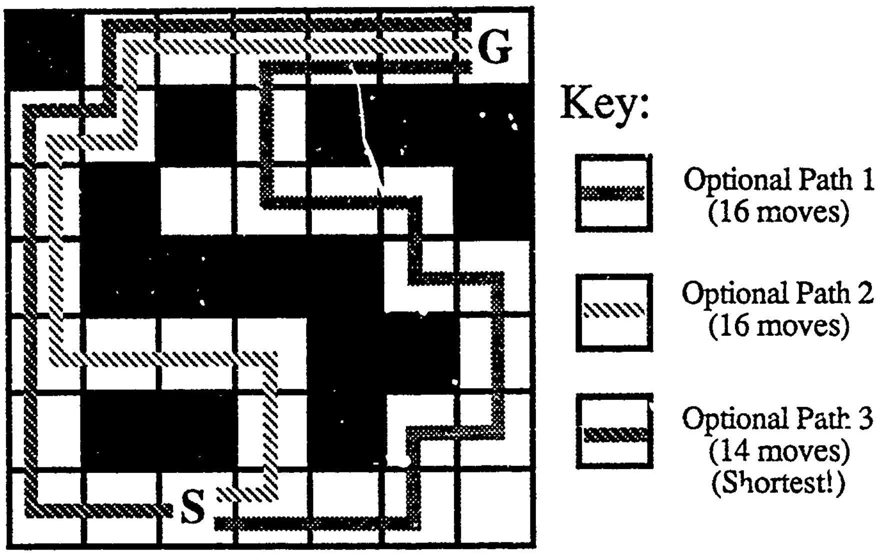

G
S
Key:
Optional Path 1 (16 moves)
Optional Path 2 (16 moves)
Optional Path 3 (14 moves) (Shortest!)

Figure 15

Analysis of AMAZ3D Non-Optimal Path Solution

	1	2	3	4	5	6	7
1	0.00	*5.01*	*5.68*	***6.35***	***7.56***	***8.78***	***10.00***
2	*3.69*	*4.35*	0.00	**5.80**	0.00	0.00	0.00
3	*3.03*	0.00	5.26	**5.27**	**4.73**	**4.21**	0.00
4	*2.38*	0.00	0.00	0.00	0.00	**3.69**	**3.17**
5	*1.73*	1.52	1.31	1.10	0.00	0.00	**2.66**
6	*1.30*	0.00	0.00	0.89	0.00	**1.91**	**2.16**
7	*0.86*	*0.43*	***Start***	**0.69**	**1.17**	**1.66**	1.91

Table 4

Voltage Potential Table for AMAZ3D Non-Optimai Test Solution

The solution path returned is Option Path 1 (16 steps, highlighted in bold in Table 4). Note however that a shorter alternative path exists, Option Path 3 (14 step, which is highlighted in italics in Table 4). The explanation for this behavior can be found by analyzing the output potentials. At row 7, col 4, there is a fork in the solution paths of Option Path 1 and

2. Due to the initial combinational effect of these to paths, the first step from the start node is east, for a voltage increase of .69, rather than west on the actual shortest path. (Note that at position (7,4) the increases are only .20 for north and .48 for the east.) This is an example of why this method can not guarantee global optimal solutions but only good solutions based on 'locally' optimal moves.

```
      1234567890123456789012345678901234567890123456789012345678901234

 1    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
 2    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
 3    XXXXX.........XXXXXXXX...................X...XXX@@@@@......XXXXX
 4    XXXXX.........XXXXXXX........G...............XXX@@@@@......XXXXX
 5    XXXXXX........XXXXXXX.........9..............XXX@@@@@.....XXXXXX
 6    XXXXXX........XXXXXXX.........8..............XXX@@@@@.....XXXXXX
 7    XXXXXX........XXXXXXX.........7..............XXX@@@@@....XXXXXXX
 8    XXXXXXX.......XXXXXXX..........6.............XXX@@@@@....XXXXXXX
 9    XXXXXXXX......XXXXXXX@.........5.............XXX@@@@@....XXXXXXX
10    XXXXXXXX......XXX@@@@@..........4............XXX@@@@@...XXXXXXXX
11    XXXXXXXXX.....XXXX@@@@...........3............XX@@@@@...XXXXXXXX
12    XXXXXXXXX.....XXX@@@@@............2..........XXX@@@@@..XXXXXXXXX
13    XXXXXXXXX.....XXXX@@@@...........X.1.........XXX@@@@@..XXXXXXXXX
14    XXXXXXXXXX....XXX@@@@@..........XXX.0........XXX@@@@@.XXXXXXXXXX
15    XXXXXXXXXX....XXXX@@@@.........XXXXX.9.......XXX@@@@@.XXXXXXXXXX
16    XXXXXXXXXXX...XXX@@@@@.........XXXXX..8......XXX@@@@@.XXXXXXXXXX
17    XXXXXXXXXXX...XXXX@@@@.........XXXXX..7......XXX@@@@@.XXXXXXXXXX
18    XXXXXXXXXXX...XXX@@@@@..........@@@...6......XXX@@@@@.XXXXXXXXXX
19    XXXXXXXXXXXX..XXX@@@@@.........@@@@...5......XXX@@@@@@XXXXXXXXXX
20    XXXXXXXXXXXX..XXX@@@@@..........@@@...4......XXX@@@@@@XXXXXXXXXX
21    XXXXXXXXXXXXX.XXX@@@@@................3......XXX@@@@@@XXXXXXXXXX
22    XXXXXXXXXXXXX.XXX@@@@@...............2........XX@@@@@@XXXXXXXXXX
23    XXXXXXXXXXXXXXXXX@@@@@..............1........XXX@@@XXXXXXXXXXXXX
24    XXXXXXXXXXXXXXXXX@@@@@.....XXX.....0.........XXX@@@XXXXXXXXXXXXX
25    XXXXXXXXXXXXXXXXX@@@@@.....XXXXX...9.........XXX@@XXXXXXXXXXXXXX
26    XXXXXXXXXXXXXXXXX@@@@@....XXXXXX...8.........XXX@@@XXXXXXXXXXXXX
27    XXXXXXXXXXXXXXXXX@@@@@....XX@@XX...7.........XXX@@XXXXXXXXXXXXXX
28    XXXXXXXXXXXXXXXXX@@@@@....XX@@@@...6.........XX@@@XXXXXXXXXXXXXX
29    XXXXXXXXXXXXXXXXXX@@@@....X@@@@@...5.........XXX@@XXXXXXXXXXXXXX
30    XXXXXXXXXXXXXXXXXX@@@@.....@@@@@...4.........XXX@@XXXXXXXXXXXXXX
31    XXXXXXXXXXXXXXXXXX@@@@......@@@....3.........XXX@@XXXXXXXXXXXXXX
32    XXXXXXXXXXXXXXXXXXX@@@.............2..........XX@@XXXXXXXXXXXXXX
33    XXXXXXXXXXXXXXXXXXX@@@.............1..........XX@XXXXXXXXXXXXXXX
34    XXXXXXXXXXXXXXXXXXX@@@.............0.........XXX@XXXXXXXXXXXXXXX
35    XXXXXXXXXXXXXXXXXXXX@@............9...........XX@XXXXXXXXXXXXXXX
36    XXXXXXXXXXXXXXXXXXXX@@............8..........XXX@XXXXXXXXXXXXXXX
37    XXXXXXXXXXXXXXXXXXXX@@...........7...........XXX@XXXXXXXXXXXXXXX
38    XXXXXXXXXXXXXXXXXXXX@@..........6............XX@XXXXXXXXXXXXXXXX
39    XXXXXXXXXXXXXXXXXXXX@@.........5..............XXXXXXXXXXXXXXXXXX
40    XXXXXXXXXXXXXXXXXXXX@@.........4.............XXXXXXXXXXXXXXXXXXX
41    XXXXXXXXXXXXXXXXXXX.@..........3.............XXXXXXXXXXXXXXXXXXX
42    XXXXXXXXXXXXXXXXXXX............2........@.X..XXXXXXXXXXXXXXXXXXX
43    XXXXXXXXXXXXXXXXXXX............1........@@XX.XXXXXXXXXXXXXXXXXXX
44    XXXXXXXXXXXXXXXXXXX............S.......@@@@@XXXXXXXXXXXXXXXXXXXX
```

Figure 16

Sample AMAZ3D 'Connected'-Obstacle Test Problem Solution

Figure 16 is an example of an outdoor terrain environment where both 'connected' and 'isolated'/ normal obstacles, as well as the eight-way moves option, have been used (see Appendix G, landnav.out). This test environment is one used by Norwood (1989) in his work with a potential field approach to navigation. Here the 'connected' obstacle nodes represent the actual obstacles (including two round obstacles in center area, and two long obstacles to the sides) and the 'isolated' obstacle nodes represent shadow regions (based on simulated laser scanner environmental input data taken on an incline rather than top down). This method is an interesting alternative which warrants further study in the future.

```
     12345678901234567890123456789012345678901234567890123456789012345678901234567890

 1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2   X........X...................X.........X...........X.................X.........X
 3   X.S1.....X...................X.........X...........X.................X.........X
 4   X...2....X...................X.........X...........D.................X.........X
 5   X....3...X...................X.........X...........X.................X.........X
 6   X.....4..X...................X.........X...........X.................X.........X
 7   XXXXXXX5XXXXXXXXXXXXXXXXXXXXXDXXXXDXXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXDXXXXXXXXXDXXX
 8   X.......67890123456789012......................................................X
 9   X........................3456..................................................X
10   X............................7.................................................X
11   X....XXXXXXXXXXXXXXXXXXXXXXXXX8........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
12   X....D..........XXX...X......X9........X........X...............X....X.........X
13   X....X..........X.....D......X0........X...XX...X..........XX...X....X.....XX..X
14   X....XX...XXXXXXX.....X......X1........X...XX...X..........XX...X....X.....XX..X
15   XXXXXXX...D.....X.....XXXXXXXX2........D...XX...X..........XX...X....D.....XX..X
16   X....D....X.....X.....X......X3........X........X..........XX...X....X.........X
17   XXX..X....X.....X.....X......X4........X........X...XXXX........X....X.........X
18   X....X....X.....X.....X......X5........X........X...............X....X....XX...X
19   XXXXXXXXXXXXXXXXXXXDXXXXXXDXXX6........XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
20   X............................7.................................................X
21   X...........................8..................................................X
22   X..........................9...................................................X
23   XXXXXXXDXXXXXXXXXXXXXXXXXXX0XXXXXXDXXXXXXXXXXXXXDXXXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
24   X........X.G..X.....X.....1..X.........X...........X.......XX........XXX.......X
25   X........X.6..X.....X....2...X.........X...........X.......XX........XXX.......X
26   X........X..5.X.....X...3....X.........X...........X.......XX........XXX.......X
27   X..X..X..X...43.....X..4.....X.........X..XX.......D.......XX........X.........X
28   X..XXXX..X....X21098765......X...XXXX..X..XX.......X...XXXXXX......XXX....XXXX.X
29   X........X....X..........XXX.X.........X..XX.......X...XXXXXX......XXX....XXXX.X
30   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Figure 17

Sample AMAZ3D Building Layout Test Problem Solution

Figure 17 shows a typical building floorplan (see Appendix G, bldgnav.out). This example shows a situation where the robot is required to travel from one room to another, through halls and doorways (unused doors are 'free spaces' shown with a 'D'). For a similar two story (3D problem) building example see Appendix G, b3dnav.out.

```
        Depth 1   Depth 2   Depth 3   Depth 4   Depth 5   Depth 6   Depth 7
        1234567   1234567   1234567   1234567   1234567   1234567   1234567

  1     432X432   5XXX5X1   6X6X6X0   7X5XXXX   8X4X2X.   XXXX3X.   87654X.
  2     XX1XXXX   XXXXXXX   XX7X7X9   XXXXXXX   9X3X1X.   XXXXXXX   9XXXXXX
  3     ..SX456   .XXXXX7   .X8X8X8   XXXX9XX   012X0X.   XXXXXXX   0123G..
  4     XXXX3XX   XXXXXXX   .X9XXXX   XXXXXXX   XXXXXX.   XXXXXXX   XXXXXX.
  5     .X2X210   .X1XXX9   .X0X678   XXXX5XX   012X4X.   XX3X3X.   654X2X.
  6     .X3XXXX   XXXXXXX   XXXXXXX   XXXXXXX   9XXXXXX   XXXXXXX   7XXX1XX
  7     .X45678   XXXXXX9   6543210   7XXXXXX   8X23456   XX1XXX7   890X098
```

Figure 18

Sample AMAZ3D 7-by-7-by-7 3D Maze Solution

As a final example, Figure 18 shows the solution to a seven-by-seven-by-seven three-dimensional test problem (see Appendix G, maz3d.out). This 3D maze is the software equivalent of a plastic Milton-Bradley 3D toy/ maze-cube which can be negotiated by a marble. (The 'cube' is in the possession of the author and I can verify that AMAZ3D's solution is correct.)

These examples show some of the capabilities of AMAZ3D. The program can be quickly reconfigured for a wide variety of navigation problems and the output can be easily modified for control of a robotic vehicle. Its only disadvantage is relatively slow path processing time. Typical computation time on a PC is on the order of two thousand node revisions per second. [Thus total time in seconds equals: (rows * columns * layers * iterations) / 2000].

Chapter 6

Conclusions

The electronic hybrid network system presented in this thesis represents an original method for constrained semi-autonomous robotic navigation control. It takes simple binary environmental input, along with a start and goal location, and processes the data through a connectionist network which provides a nodal 'voltage potential' look-up table. The voltage potentials are analyzed by a second network which determines the move direction based on an examination of the neighborhood around a given current node (beginning with the start position). Finally, the system presents a solution path based on a set of locally optimal steps.

This system exhibits some distinct advantages over the traditional approaches noted earlier. Due to the parallel architecture of the connectionist network, the system can be expected to be much faster (and possibly more damage resistant) than Artificial Intelligence search algorithms. This navigation system also has the flexibility to account for both moving obstacles and a moving goal. This is accomplished by simply applying new inputs to the connectionist network and reevaluating the path problem. Also, assuming the hybrid network system is implemented in hardware, it does not require a dedicated CPU/ microcomputer to make it work and could conceivably be built right into the sensory system and servomotors of a robot.

An example of a near term use for an expanded 'Maze Machine'-type navigation system is the control of a robotic delivery vehicle in a large factory. The environment/ floorplan would be fairly stable, however local sensors in the factory could easily update the 'maze' database on board the robot through radio communications. A human or central computer would assign the robot a task, probably also by radio link. An example task could be to "take pallet #196 from point A to point B". The navigation system on board the vehicle would then plan the path from the current location to point A (the pick up point for the pallet) and then plan a second path from A to B (the pallet drop off point). Other routines on board would handle the pallet upload/ download procedure, emergency stop procedures, etc.

There are disadvantages as well. First, the traditional AI procedures, regardless of whether they find an optimal path or just feasible paths, are proven methods that can be implemented relatively cheaply on microcomputers. The network system presented here can not be implemented on microcomputers while keeping its parallel architecture, however the sequential simulation AMAZ3D can be a valuable alternative method, especially in feasibility tests to determine if a custom VLSI network setup is warranted for a particular application. Since the AI approaches are software based, they can be modified much more easily to represent different type/ size systems and environments. Also, because they are tried and proven procedures, software is readily available for their implementation.

In summary, this electronic hybrid connectionist network system solves path planning/ obstacle avoidance problems for a grid-structured two- or three-dimensional environment. This navigation system provides good (often optimal) path solutions based on a collection of locally optimal steps/ moves found using the output of the sensory analysis connectionist network. It operates using only low-level (binary) descriptions of the environment which can be provided by a variety of current and experimental sensory systems. The navigation machine would best be used in combination with a computer and global sensory input systems. Possible applications of this hybrid network system span from the home to industry and even to outer space.

Chapter 7

Areas for Future Work

The work already done on the development of this hybrid connectionist network system opens many possibilities for future efforts to expand this project and provides the potential for other developments. One very useful development would be an interface for input and output between a microcomputer and an expanded hardware 'Maze Machine' network device. An efficient system is needed for updating the binary inputs of the connectionist network, as well as for quickly taking the path output and formating it as robotic movement instructions. This interface would replace the plugs used to establish a particular maze initialization on the Maze Machine, the switches used to control external input, and the LEDs used for the output displays. (Figure A9, Appendix A, shows an improved system which handles the functions of Modules 1 and 2 of the Maze Machine and also lends itself to a computer interface.) Another extension of the research is the use of varied resistances and single direction links in the connectionist network. These modifications would be used to simulate cross-country travel across rough terrain (i.e. going uphill and downhill on varied slopes) and travel on city streets (where resistances represent speed limits and one-way streets are modeled using diodes/ direction dependent connections). Testing the navigation system, using an actual sensory system and robotic vehicle, is also desirable for greater whole system credibility.

The most obvious extension is the development of a single analog VLSI chip to represent a modular electronic hybrid system (thus replacing the 65 current 'off-the-shelf' CMOS IC chips used in this project). A modular electronic system could then be expanded to represent much larger two- and three-dimensional environments. An alternative to building a complete 'Maze Machine' on-a-chip would be to provide a combination hardware/ software interactive system by developing a plug-in VLSI based connectionist network board for microcomputers; the computer would be required to do the auxiliary work but the system would still take advantage of the parallel processing structure of the sensory analysis connectionist network and thus almost instantaneously solve the numerous simultaneous linear equations modeled by this method. Finally, this system could be developed for commercial use. Such uses could include wheelchair control to help the handicapped navigate through a known environment (such as their house) or a robotic parts delivery system in a factory or warehouse. The possibilities are limited only by our imagination.

Bibliography

Adnan, S., and Cheatham, J. B.Jr. (1990), "An Omnidirectional Platform to Simulate a Free-Flying Robot," *ISMCR '90, First International Symposium on Measurement and Control in Robotics*, Houston, TX, June 1990.

Alexander, R. S., and Rowe, N. C. (1990), "Path Planning by Optimal-Path-Map Construction for Homogeneous-Cost Two-Dimensional Regions," *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 1924-1929, May 1990.

Anderson, J. A., and Rosenfeld, E. (Eds.) (1988), *Neurocomputing: Foundations of Research*, MIT Press: Cambridge, MA.

Badreddin, E. (1990), "Associative Memory Implementation in Path-Planning for Mobile Robots," *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 14-19, May 1990.

Cheatham, J. B.Jr., Regalbuto, M. A., Krouskop, T. A., and Winningham, D. J. (1987), "A Mobile Robotic System as an aid for the Severely Handicapped," Proc. 9th IEEE/EMBS Conf., Boston, MA.

Cheatham, J. B.Jr., and Adnan, S. (1989), "Kinematic Analysis and Trajectory Control of a Mobile Omni-directional Robot," *First National Applied Mechanisms & Robotics Conference*, Cincinnati, Ohio, November 1989.

Craig, J. J. (1986), *Introduction to Robotics: Mechanics and Control*, Addison-Wesley: Reading, MA.

Fu, K. S., Gonzalez, R. C., and Lee, C. S. G. (1987), *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill: New York, NY.

Gat, E., Slack, M. G., Miller, D. P., and Firby, R. J. (1990), "Path Planning and Execution Monitoring for a Planetary Rover," *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 20-25, May 1990.

Graf, H. P., Jackel, L. D., and Hubbard, W. E. (1988), "VLSI Implementation of a Neural Network Model," *Computer*, March 1988, pp. 41-49.

Grossberg, S. (1988), "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks*, vol. 1, pp. 17-61.

Hecht-Nielson, R. (1988), "Neurocomputing: Picking the Human Brain," *IEEE Spectrum*, vol. 25, pp. 36-41.

Hopfield, J. J., and Tank, D. W. (1985), "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, pp. 141-152.

Hutchinson, J., Koch, C., Luo, J., and Mead, C. (1988), "Computing Motion Using Analog and Binary Resistive Networks," *Computer*, March 1988, pp. 52-63.

Koch, C. (1987), "Computing Motion in the Presence of Discontinuities: Algorithm and Analog Networks," *Proceeding of the NATO Advanced Research Workshop on Neural Computers*, Neuss, F.R.G., September 1987, pp. 101-110, Springer-Verlag: Berlin, W. Germany.

Lancaster, D. (1989), *CMOS Cookbook*, 2nd Ed., H. W. Sams & Co: Indianapolis, IN.

Lippman, R. P. (1987), "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22.

McClelland, J. L., and Rumelhart, D. E. (1988), *Explorations in Parallel Distributed Processing*, MIT Press: Cambridge, MA.

Mead, C. (1989), *Analog VLSI and Neural Systems*, Addison-Wesley: Reading, MA.

Mitchell, J. S. B. (1988), "An Algorithmic Approach to Some Problems in Terrain Navigation," *Artificial Intelligence, Special Volume on Geometric Reasoning*, vol. 37 numbers 1-3, pp. 171-201.

Moopenn, A., and Thakoor, A. P. (1989), "Programmable Synaptic Devices for Electronic Neural Nets," *Proceedings of the 5th IASTED International Conference on Expert Systems and Neural Networks*, Honolulu, HI.

Norwood, J. D. (1989), "Robotic Path Planning and Obstacle Avoidance: A Neural Network Approach," Master of Science Thesis in Mechanical Engineering, Rice University.

Ögmen, H. (1989), "Neural Networks Lecture Notes," ELEE 6397 - Neural Networks, University of Houston, Fall 1989.

Regalbuto, M. A., Fisher, P. B., Adnan, S., Norwood, J. D., and Weiland, P. L. (1988), "A Navigation System Framework for a Mobile Robot," Proc. 10th IEEE/EMBS Conf., New Orleans, LA.

Regalbuto, M. A. (1990), "A Semi-Autonomous Mobile Robot/ Teleoperator With Applications as an Aid for Severely Handicapped People," Ph. D. Dissertation in Mechanical Engineering, Rice University.

Rich, E. (1983), *Artificial Intelligence*, McGraw-Hill: New York, NY.

Rietman, E. (1988), *Experiments in Artificial Neural Networks*, Tab: Blue Ridge Summit, PA.

Shiller, Z., and Chen, J. C. (1990), "Optimal Motion Planning of Autonomous Vehicles in Three Dimensional Terrains," *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 198-203, May 1990.

Shiva, S. G. (1988), *Artificial Intelligence*, Scott, Foresman and Company: Glenview, IL.

Smith, R. J. (1976), *Circuits Devices and Systems*, 3rd Ed., Wiley: New York, NY.

Staugaard, A. C. Jr. (1987), *Robotics and AI*, Prentice Hall: Englewood Cliffs, NJ.

Tank, D. W. , and Hopfield, J. J. (1987), "Collective Computation in Neuronlike Circuits," *Scientific American*, vol. 257, pp. 104-114.

Tilove, R. B. (1990), "Local Obstacle Avoidance for Mobile Robots Based on the Method of Artificial Potentials," *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 566-571, May 1990.

Tokuta, A., and Hughes, K. (1990), "Scanline Algorithms in Robot Path Planning," *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 192-197, May 1990.

Wakerly, J. F. (1976), *Logic Design Projects Using Standard Integrated Circuits*, Wiley: New York, NY.

Wasserman, P. D. (1989), *Neural Computing: Theory and Practice*, Van Nostrand Reinhold: New York, NY.

Weiland, P. L. (1989), "The Use of Scanning Laser Devices for Autonomous Robotic Navigation," Master of Science Thesis in Mechanical Engineering, Rice University.

Winston, P. H. (1984), *Artificial Intelligence*, 2nd Ed., Addison-Wesley: Reading, MA.

Wolovich, W. A. (1987), *Robotics: Basic Analysis and Design*, Holt, Rinehart and Winston: New York, NY.

Wu, C. K. (1989), "Laser Imaging Simulation System," Ph. D. Dissertation in Mechanical Engineering, Rice University.

Yoh-Han Pao (1989), *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley: Reading, MA.

Appendix A

Maze Machine Electronic Module Diagrams

The following electronic diagrams show the structures of the six modules that, when properly inter-connected, make up the Maze Machine. For more detailed wiring connection information see Appendix B, Maze Machine Module Wiring Tables. Notes for each module are as follows:

Module 1 - Input) Consists of hand-placed electrically-wired 'plugs' (see Figure A1) which fit into sockets of Module 2. Four kinds of plugs are used to represent the four possible conditions of a node: obstacle, free space, goal or start. The wiring pattern for each of the nodes results in:

1) **Obstacle node** voltage outputs (also the outer boundary) are forced to GND and the links to its neighbors are disconnected.
2) **Free space node** voltage outputs are allowed to 'settle' on the average value of the neighbor nodes influencing it (recurrent behavior).
3) **Goal node** voltage output is forced to VCC (source/ supply voltage) and this voltage potential is allowed to influence the neighboring nodes.
4) **Start node** voltage output is forced to GND and this voltage potential is allowed to influence the neighboring nodes.

Module 2 - Connectionist Network) This network module (see Figures A2 and A3) consists of 49 wired sockets interconnected by 100 kΩ resistors as shown in the diagrams. These node sockets receive binary inputs from the Module 1 plugs and output 'voltage potentials' (analog) derived as a function of the various external inputs and the interconnections between neighboring nodes. Note that no pins within a node socket are connected until a plug is inserted.

Module 3 - Multiplexer) This module (see Figures A4 and A5) multiplexes the 49 analog signals from Module 2 (using current node address from Module 5) and outputs five analog values. The first four lines carry the 'voltage potentials' of the four nearest neighbors of the current node and the fifth provides the current node's 'voltage potential' as a separate (external) output for sampling.

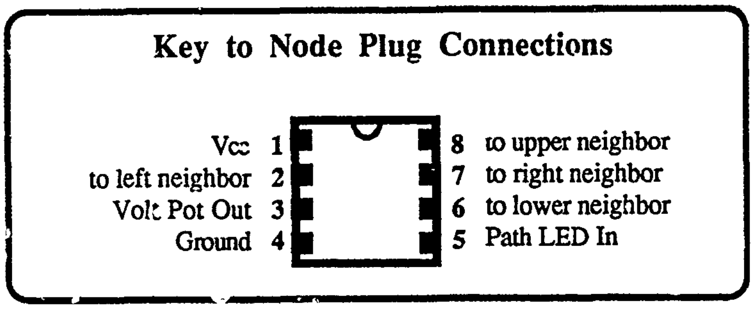
Module 4 - Local-Move-Finder Network) This network module (see Figure A6) takes input from Module 3 and outputs four binary lines to Module 5. It provides the next-move direction (of the four possibilities), using the greatest local voltage increase as the decision criteria.

Module 5 - Move Control) This module (see Figure A7) provides the cycle control which steps the machine through the desired navigation path, one move at a time. It provides the current location address to Module 3 and updates this BCD location based on Module 4's output. This module can be provided with external inputs for manually setting the current address to any desired value for running special tests as desired (see Appendix D, Procedure for Running Maze Machine Auto-Path Test). This circuit also sends the current address to Module 6. Not shown are two BCD to 7-segment LED displays used to indicate the current row and column and 4 LEDs used to indicate the local move direction.

Module 6 - Path Output Display) This Module (see Figure A8) contains multiplexing circuitry and 49 RS flip-flops, one per node, used to appropriately light an array of 49 LEDs which display the solution path (and the intermediate moves as the path is being generated).

Figure A9 - Diagram of Alternate Input/ Connectionist Network Node Structure (To Replace Modules 1 and 2) is included to show an alternative hardware method which can replace the 'awkward' plug and socket system of Modules 1 and 2. The sketch shows the 3 IC chips needed for each node.

Maze Machine
Module 1 Diagram - Input



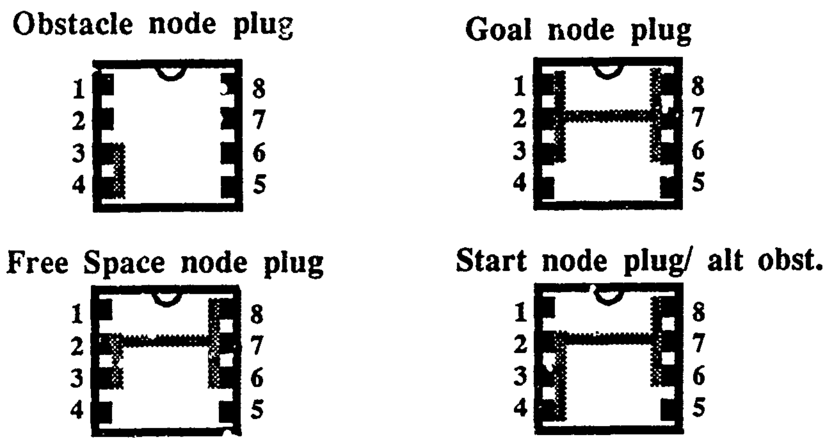


Figure A1

Module 1 Diagram - Input

Maze Machine
Module 2 Diagram - Connectionist Network

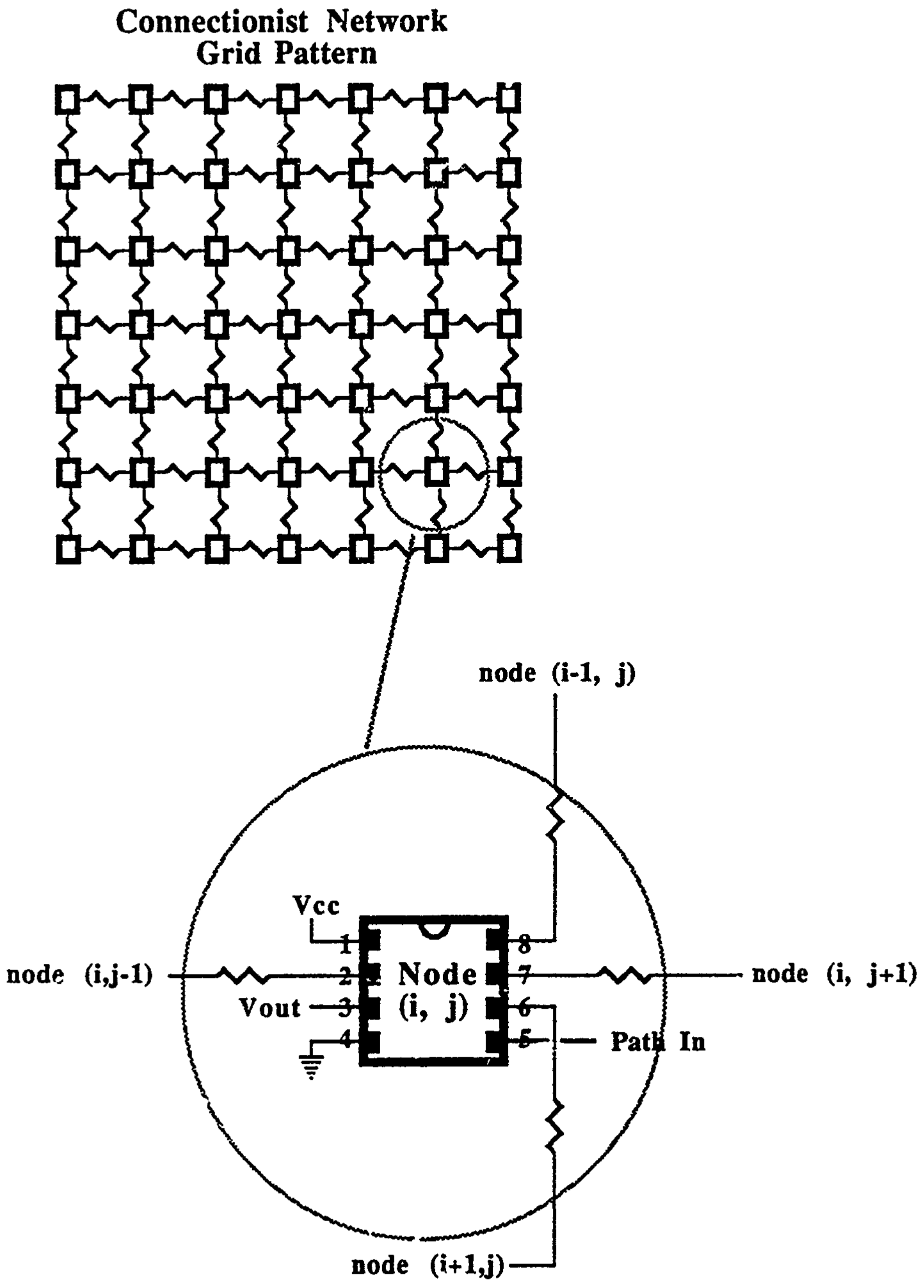


Figure A2

Module 2 Diagram - Connectionist Network

Maze Machine
Module 2 Diagram - Connectionist Network continued

Partial Wiring Diagram for nodes 1, 2, 8, 9, 15, & 16

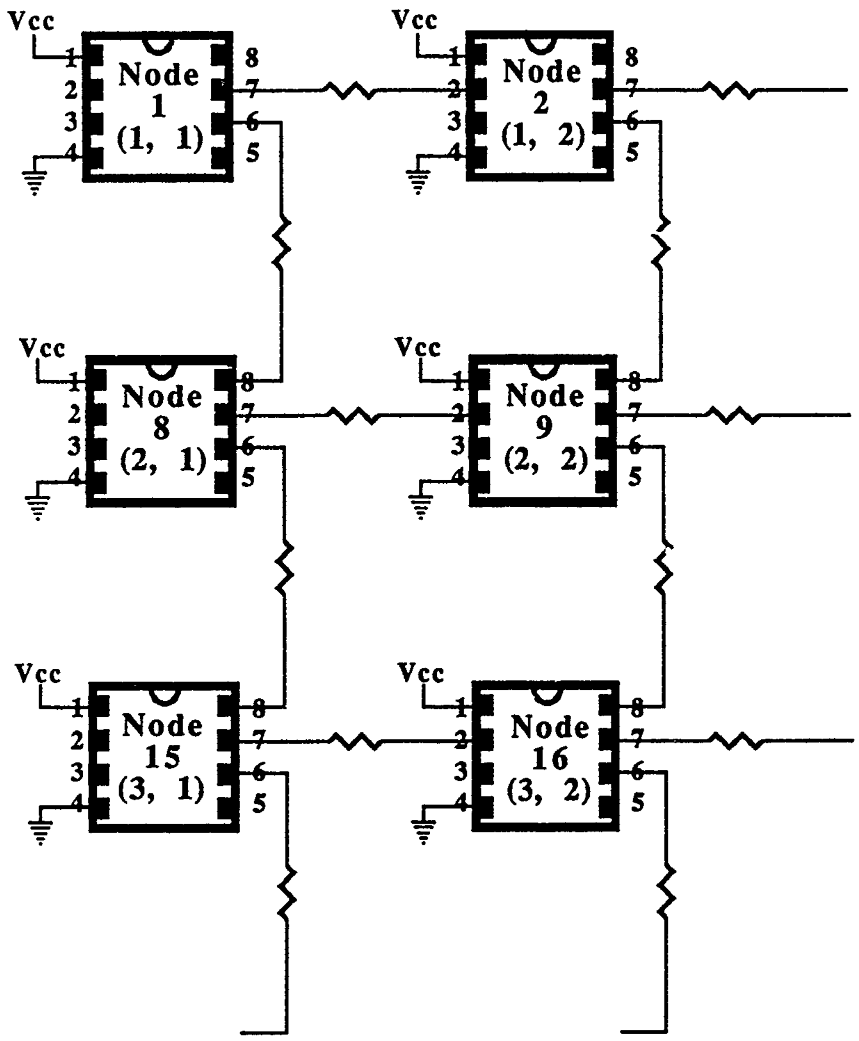


Figure A3

Module 2 Diagram - Connectionist Network continued

Maze Machine
Module 3 Diagram - Multiplexer

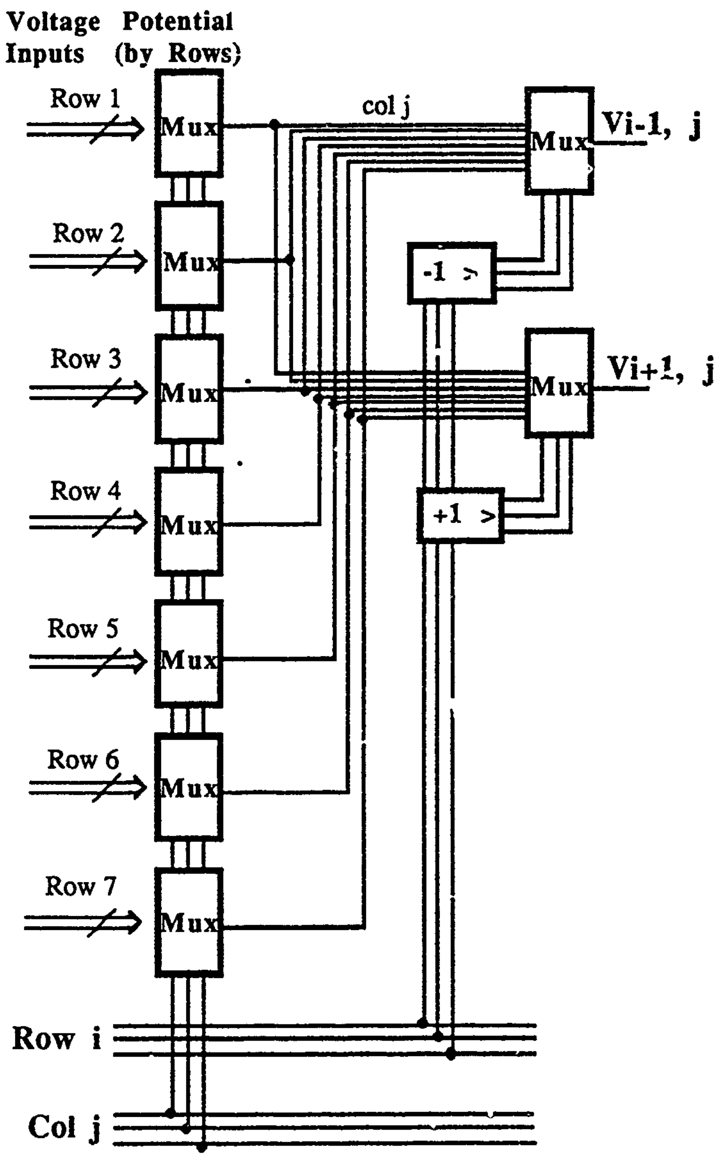


Figure A4

Module 3 Diagram - Multiplexer

Maze Machine
Module 3 Diagram - Multiplexer continued

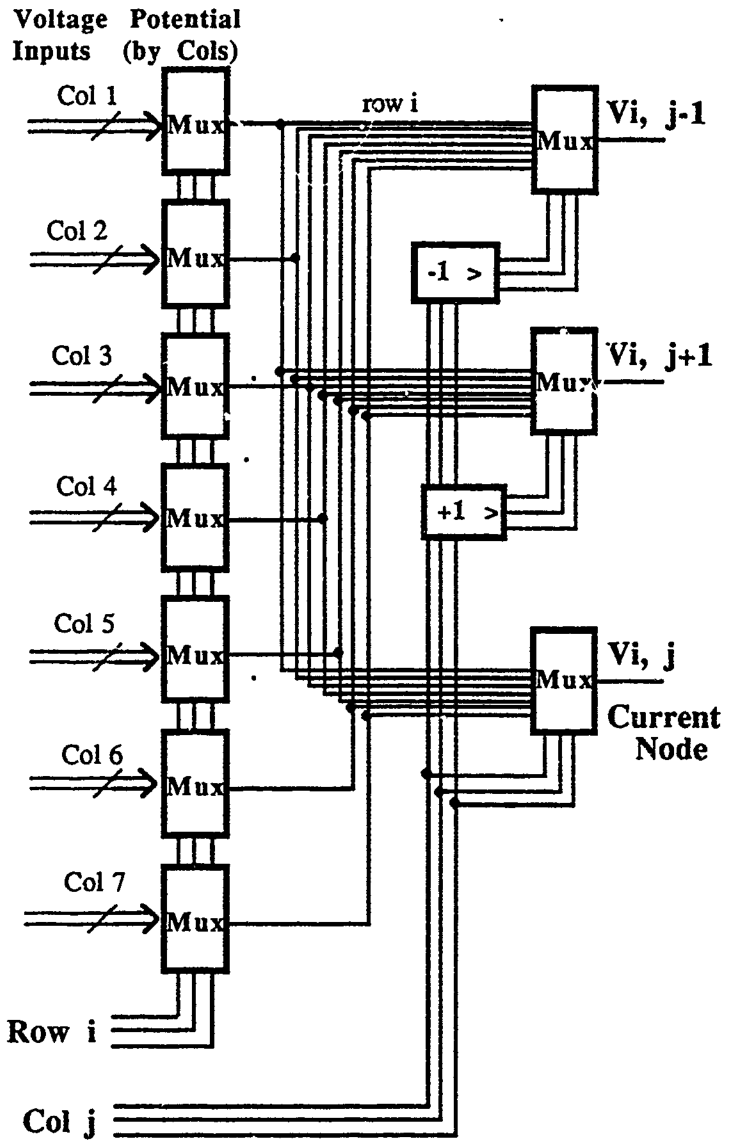


Figure A5

Module 3 Diagram - Multiplexer continued

Maze Machine
Module 4 Diagram - Local-Move-Finder Network

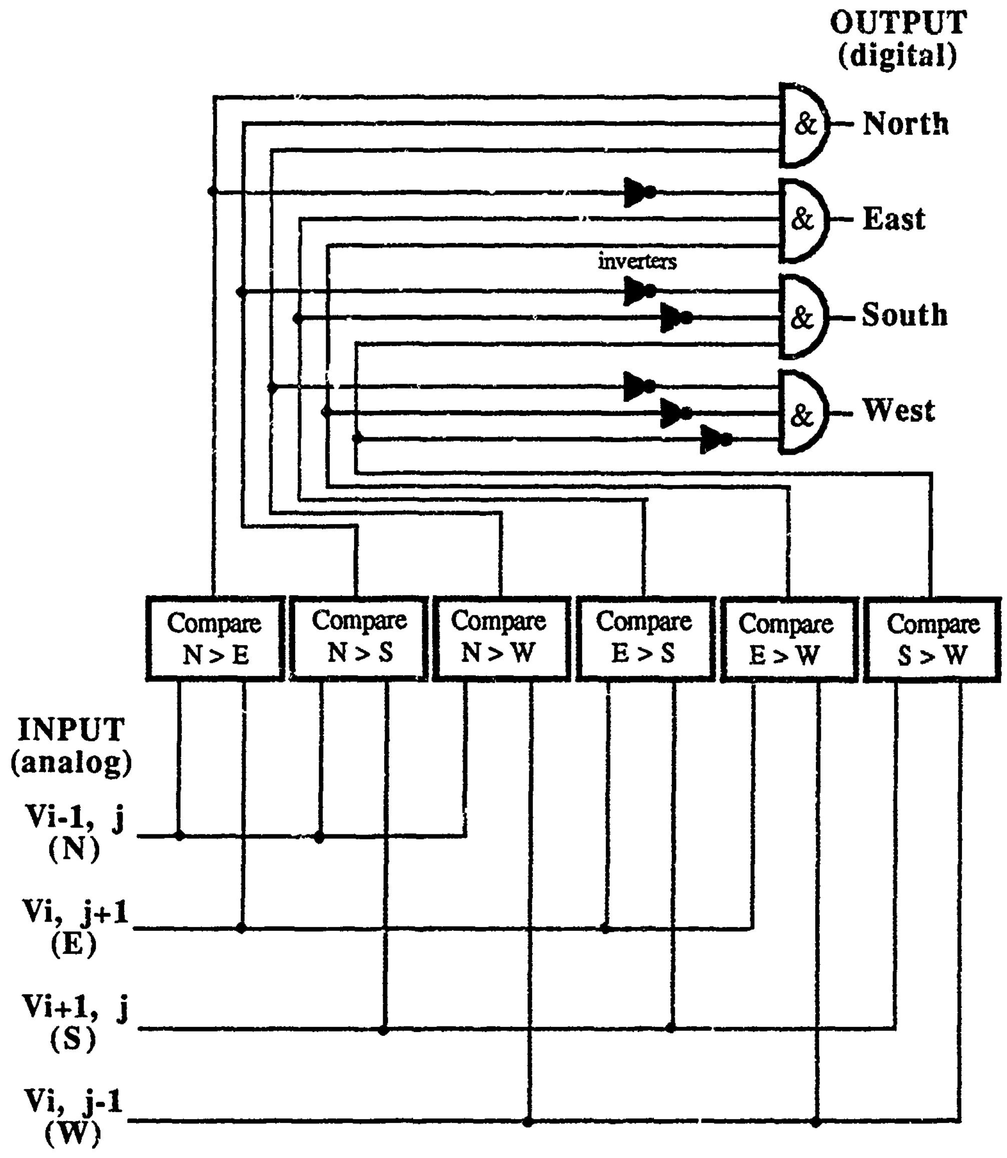


Figure A6

Module 4 Diagram - Local-Move-Finder Network

Maze Machine
Module 5 Diagram - Move Control

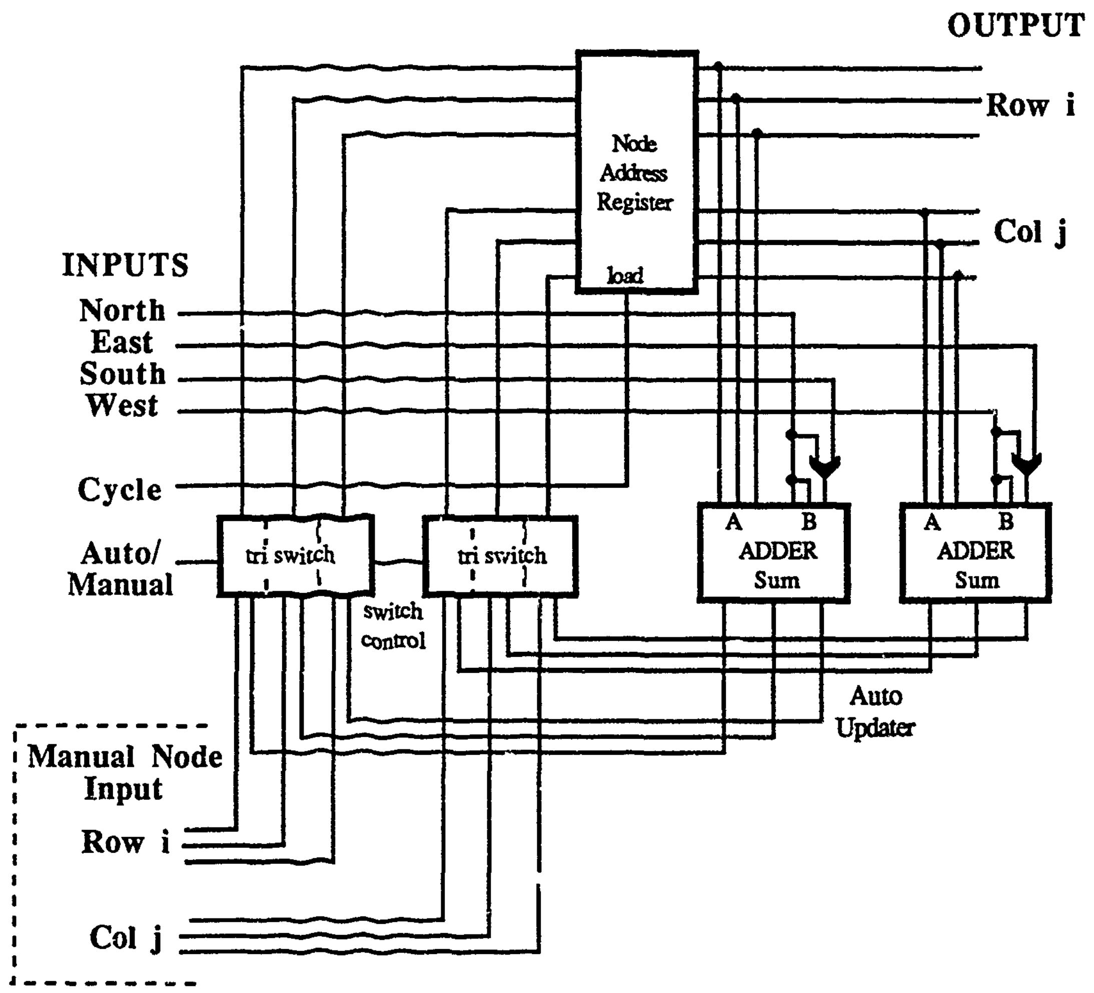


Figure A7

Module 5 Diagram - Move Control

Maze Machine
Module 6 Diagram - Path Output Display

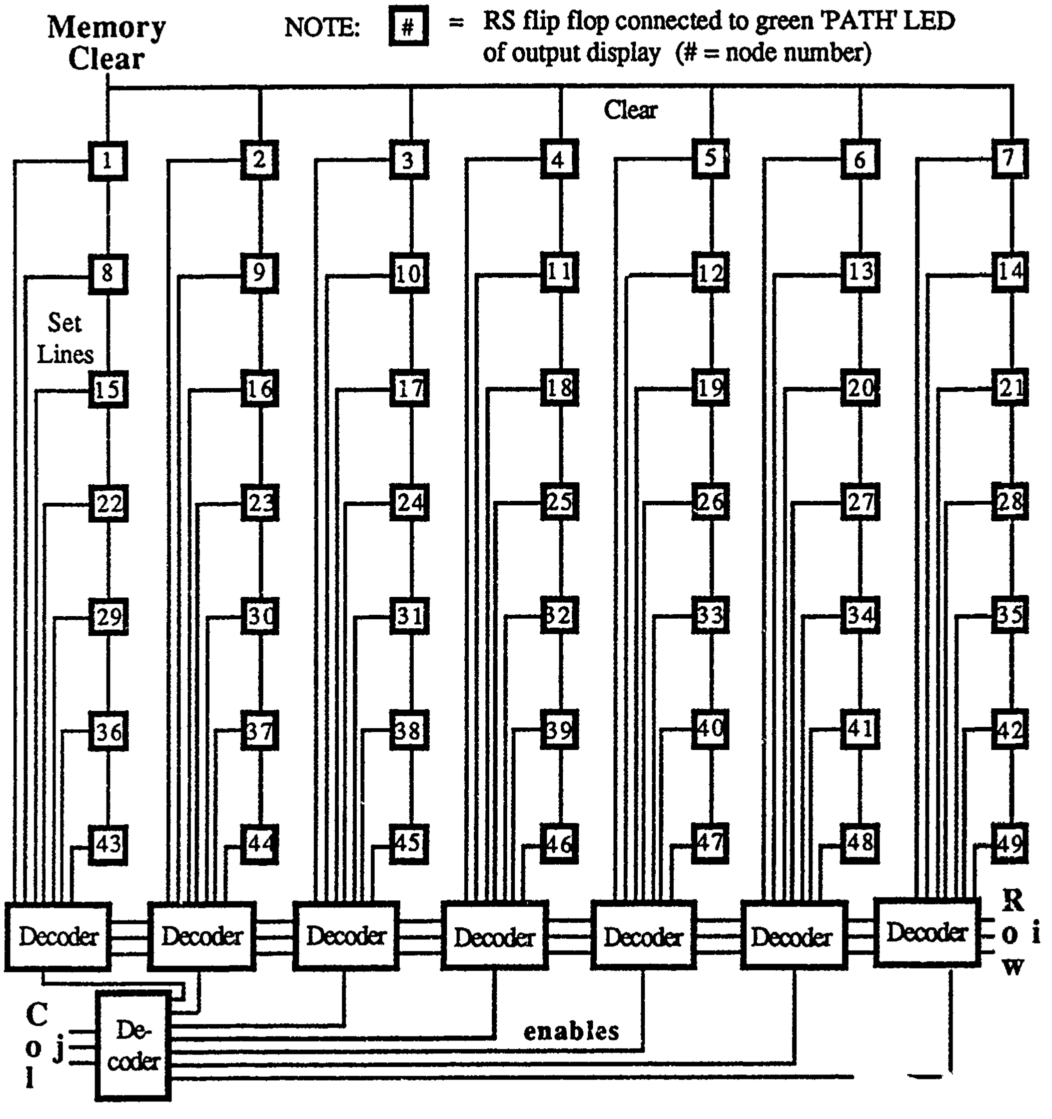


Figure A8

Module 6 Diagram - Path Output Display

Diagram of Alternate Input/ Connectionist Network Node Structure (To Replace Modules 1 and 2)

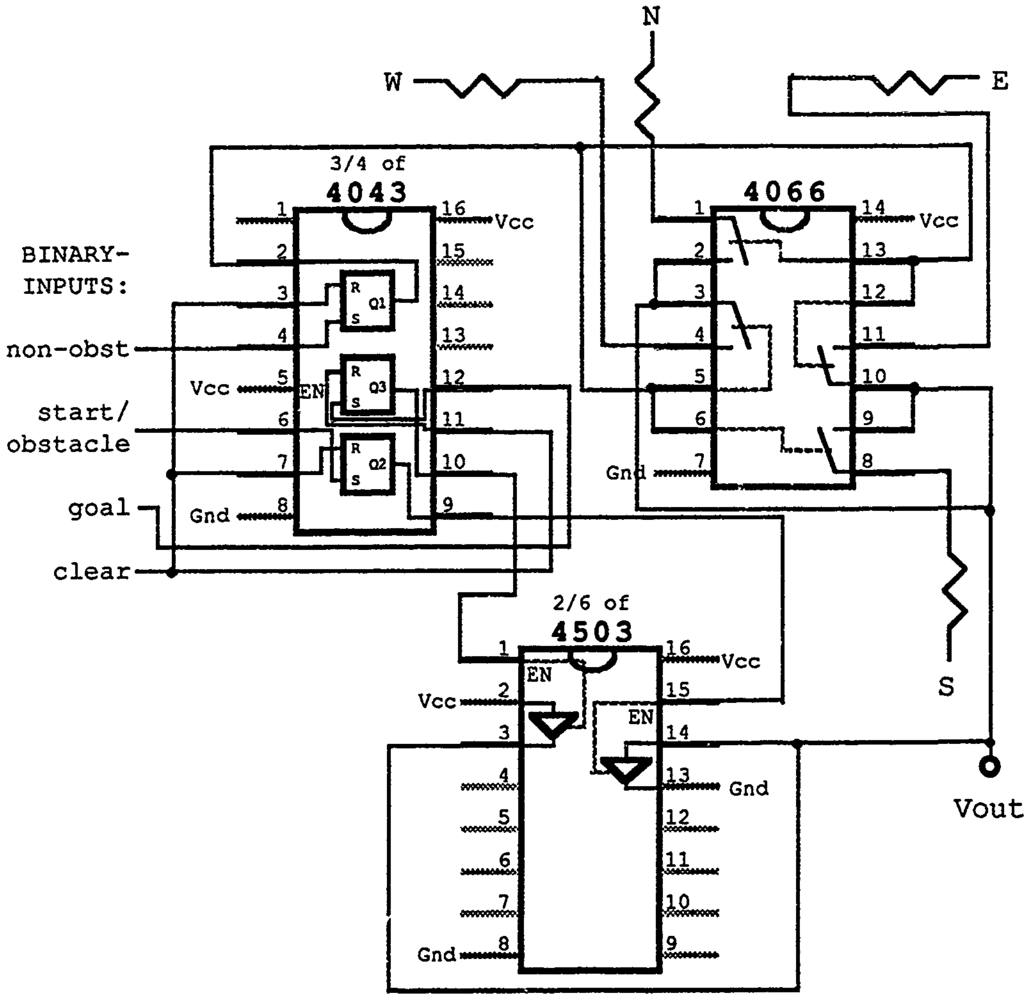


Figure A9

Diagram of Single Node for Alternate Method to Replace Modules 1 and 2

Appendix B

Maze Machine Electronic Module Wiring Tables

The following wire wrapping tables are grouped by Maze Machine Module and comprise the complete list of connections needed to construct the machine. Wire wrapping was used rather than bread-boarding or soldering in order to keep the size down and simplify to the highly interconnected customized design, which incorporated large numbers of IC chip pin connections that had multiple routings to other chips. The wire wrapping took Bill Atkinson (see Acknowledgements) and myself more than a month to complete and debug.

NOTES:

1) Module 6's chip numbering does not start with 1, but rather 32, due to my use of a surplus 60 chip-socket custom wire wrap board which was prenumbered and used to house both Modules 3 and 6. (The pre-printed numbers on the board were used.)

2) There are several individual 'chip' tables, as well as destination entries in other tables, which refer to Data Bus(ses) and Resistor Pack. This format was used due to its simplicity, as well as for the fact that actual chip sockets and compatible jumper cables were used to connect the modules (and hold the needed resistors in some cases).

Module #3 - MULTIPLEXER - Wiring Table B1

#	1	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	4	node
2	6	OUT2	6	node
3	I/O	8,9	14	1
4	7	OUT2	7	node
5	5	OUT2	5	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	3	node
13	0	x	x	Gnd
14	1	OUT2	1	node
15	2	OUT2	2	node
16	Vcc	x	x	Vcc

#	2	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	11	node
2	6	OUT2	13	node
3	I/O	8,9	15	2
4	7	OUT2	14	node
5	5	OUT2	12	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	10	node
13	0	x	x	Gnd
14	1	OUT2	8	node
15	2	OUT2	9	node
16	Vcc	x	x	Vcc

#	3	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	18	node
2	6	OUT2	20	node
3	I/O	8,9	12	3
4	7	OUT2	21	node
5	5	OUT2	19	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	17	node
13	0	x	x	Gnd
14	1	OUT2	15	node
15	2	OUT2	16	node
16	Vcc	x	x	Vcc

#	4	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	25	node
2	6	OUT2	27	node
3	I/O	8,9	1	4
4	7	OUT2	28	node
5	5	OUT2	26	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	24	node
13	0	x	x	Gnd
14	1	OUT2	22	node
15	2	OUT2	23	node
16	Vcc	x	x	Vcc

#	5	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	32	node
2	6	OUT2	34	node
3	I/O	8,9	5	5
4	7	OUT2	35	node
5	5	OUT2	33	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	31	node
13	0	x	x	Gnd
14	1	OUT2	29	node
15	2	OUT2	30	node
16	Vcc	x	x	Vcc

#	6	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	39	node
2	6	OUT2	41	node
3	I/O	8,9	2	6
4	7	OUT2	42	node
5	5	OUT2	40	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	38	node
13	0	x	x	Gnd
14	1	OUT2	36	node
15	2	OUT2	37	node
16	Vcc	x	x	Vcc

#	7	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	46	node
2	6	OUT2	48	node
3	I/O	8,9	4	7
4	7	OUT2	49	node
5	5	OUT2	47	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	OUT2	45	node
13	0	x	x	Gnd
14	1	OUT2	43	node
15	2	OUT2	44	node
16	Vcc	x	x	Vcc

#	8	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	4	3	node
2	6	6	3	node
3	I/O	DB	1	VN
4	7	7	3	node
5	5	5	3	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	10	12	row4
10	B/2	10	11	row2
11	A/1	10	10	row1
12	3	3	3	node
13	0	x	x	Gnd
14	1	1	3	node
15	2	2	3	node
16	Vcc	x	x	Vcc

#	9	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	4	3	node
2	6	6	3	node
3	I/O	DB	3	VS
4	7	7	3	node
5	5	5	3	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	11	12	row4
10	B/2	11	11	row2
11	A/1	11	10	row1
12	3	3	3	node
13	0	x	x	Gnd
14	1	1	3	node
15	2	2	3	node
16	Vcc	x	x	Vcc

Module #3 - MULTIPLEXER - Wiring Table B1, continued

#	10	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A4	x	x	Gnd
2	B3	x	x	Vcc
3	A3	DB	12	row4
4	B2	x	x	Vcc
5	A2	DB	13	row2
6	B1	x	x	Vcc
7	A1	DB	14	row1
8	Gnd	x	x	Gnd
9	CI	x	x	Gnd
10	S1	8	11	row1
11	S2	8	10	row2
12	S3	8	9	row4
13	S4	x	x	open
14	CO	x	x	open
15	B4	x	x	Vcc
16	Vcc	x	x	Vcc

#	11	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A4	x	x	Gnd
2	B3	x	x	Gnd
3	A3	DB	12	row4
4	B2	x	x	Gnd
5	A2	DB	13	row2
6	B1	x	x	Gnd
7	A1	DB	14	row1
8	Gnd	x	x	Gnd
9	CI	x	x	Vcc
10	S1	9	11	row1
11	S2	9	10	row2
12	S3	9	9	row4
13	S4	x	x	open
14	CO	x	x	open
15	B4	x	x	Gnd
16	Vcc	x	x	Vcc

#	12	DATA BUS (DB)		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	VN	OUT3	x	VN
2	VE	OUT3	x	VE
3	VS	OUT3	x	VS
4	VW	OUT3	x	VW
5	N	OUT4	x	N
6	E	OUT4	x	E
7	S	OUT4	x	S
8	W	OUT4	x	W
9	col4	OUT5	x	col4
10	col2	OUT5	x	col2
11	col1	OUT5	x	col1
12	row4	OUT5	x	row4
13	row2	OUT5	x	row2
14	row1	OUT5	x	row1
15	Gnd	x	x	Gnd
16	Vcc	x	x	Vcc

#	13	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	22	node
2	6	OUT2	36	node
3	I/O	20,21	14	1
4	7	OUT2	43	node
5	5	OUT2	29	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	15	node
13	0	x	x	Gnd
14	1	OUT2	1	node
15	2	OUT2	8	node
16	Vcc	x	x	Vcc

#	14	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	23	node
2	6	OUT2	37	node
3	I/O	20,21	15	2
4	7	OUT2	44	node
5	5	OUT2	30	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	16	node
13	0	x	x	Gnd
14	1	OUT2	2	node
15	2	OUT2	9	node
16	Vcc	x	x	Vcc

#	15	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	24	node
2	6	OUT2	38	node
3	I/O	20,21	12	3
4	7	OUT2	45	node
5	5	OUT2	31	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	17	node
13	0	x	x	Gnd
14	1	OUT2	3	node
15	2	OUT2	10	node
16	Vcc	x	x	Vcc

#	16	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	25	node
2	6	OUT2	39	node
3	I/O	20,21	1	4
4	7	OUT2	46	node
5	5	OUT2	32	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	18	node
13	0	x	x	Gnd
14	1	OUT2	4	node
15	2	OUT2	11	node
16	Vcc	x	x	Vcc

#	17	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	26	node
2	6	OUT2	40	node
3	I/O	20,21	5	5
4	7	OUT2	47	node
5	5	OUT2	33	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	19	node
13	0	x	x	Gnd
14	1	OUT2	5	node
15	2	OUT2	12	node
16	Vcc	x	x	Vcc

#	18	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	27	node
2	6	OUT2	41	node
3	I/O	20,21	2	6
4	7	OUT2	48	node
5	5	OUT2	34	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	20	node
13	0	x	x	Gnd
14	1	OUT2	6	node
15	2	OUT2	13	node
16	Vcc	x	x	Vcc

Module #3 - MULTIPLEXER - Wiring Table B1, continued

#	19	4051 1 of 8 swich		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	OUT2	28	node
2	6	OUT2	42	node
3	I/O	20,21	4	7
4	7	OUT2	49	node
5	[illegible]	OUT2	35	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	12	row4
10	B/2	DB	13	row2
11	A/1	DB	14	row1
12	3	OUT2	21	node
13	0	x	x	Gnd
14	1	OUT2	7	node
15	2	OUT2	14	node
16	Vcc	x	x	Vcc

#	20	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	16	3	node
2	6	18	3	node
3	I/O	DB	4	VW
4	7	19	3	node
5	5	17	3	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	22	12	col4
10	B/2	22	11	col2
11	A/1	22	10	col1
12	3	15	3	node
13	0	x	x	Gnd
14	1	13	3	node
15	2	14	3	node
16	Vcc	x	x	Vcc

#	21	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	16	3	node
2	6	18	3	node
3	I/O	DB	2	VE
4	7	19	3	node
5	5	17	3	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	23	12	col4
10	B/2	23	11	col2
11	A/1	23	10	col1
12	3	15	3	node
13	0	x	x	Gnd
14	1	13	3	node
15	2	14	3	node
16	Vcc	x	x	Vcc

#	22	4008 4 bit F Adder		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A4	x	x	Gnd
2	B3	x	x	Vcc
3	A3	DB	9	col4
4	B2	x	x	Vcc
5	A2	DB	10	col2
6	B1	x	x	Vcc
7	A1	DB	11	col1
8	Gnd	x	x	Gnd
9	CI	x	x	Gnd
10	S1	20	11	col1
11	S2	20	10	col2
12	S3	20	9	col4
13	S4	x	x	open
14	CO	x	x	open
15	B4	x	x	Vcc
16	Vcc	x	x	Vcc

#	23	4008 4 bit F Adder		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A4	x	x	Gnd
2	B3	x	x	Gnd
3	A3	DB	9	col4
4	B2	x	x	Gnd
5	A2	DB	10	col2
6	B1	x	x	Gnd
7	A1	DB	11	col1
8	Gnd	x	x	Gnd
9	CI	x	x	Vcc
10	S1	21	11	col1
11	S2	21	10	col2
12	S3	21	9	col4
13	S4	x	x	open
14	CO	x	x	open
15	B4	x	x	Gnd
16	Vcc	x	x	Vcc

#	24	4051 1 of 8 swtch		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	4	16	3	node
2	6	18	3	node
3	I/O	Out3sp	x	Vnde
4	7	19	3	node
5	5	17	3	node
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	C/4	DB	9	col4
10	B/2	DB	10	col2
11	A/1	DB	11	col1
12	3	15	3	node
13	0	x	x	Gnd
14	1	13	3	node
15	2	14	3	node
16	Vcc	x	x	Vcc

Module #4 - LOCAL-MOVE-FINDER NETWORK - Wiring Table B2

#	1	339 quad comp		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Out2	3\4	2\3	Comp2
2	Out1	3\4	1\1	Comp1
3	Vcc	x	x	Vcc
4	In1-	DB	2	VE
5	In1+	DB	1	VN
6	In2-	DB	3	VS
7	In2+	same	5	VN
8	In3-	DB	4	VW
9	In3+	same	5	VN
10	In4-	same	6	VS
11	In4+	same	4	VE
12	Gnd	x	x	Gnd
13	Out4	3\4	4\5	Comp4
14	Out3	3\4	8\9	Comp3

#	2	339 quad comp		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Out2	3\4	13\13	Comp6
2	Out1	3\4	5\11	Comp5
3	Vcc	x	x	Vcc
4	In1-	DB	4	VW
5	In1+	DB	2	VE
6	In2-	same	4	VW
7	In2+	DB	3	VS
8	In3-	x	x	Gnd
9	In3+	x	x	Gnd
10	In4-	x	x	Gnd
11	In4+	x	x	Gnd
12	Gnd	x	x	Gnd
13	Out4	x	x	open
14	Out3	x	x	open

#	3	4073 tri 3 AND		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	1	2	Comp1
2	In1b	1	1	Comp2
3	In2a	4	2	Icmp1
4	In2b	1	13	Comp4
5	In2c	2	2	Comp5
6	Out2	DB	6	E
7	Gnd	x	x	Gnd
8	In1c	1	14	Comp3
9	Out1	DB	5	N
10	Out3	DB	7	S
11	In3a	4	4	Icmp2
12	In3b	4	6	Icmp4
13	In3c	2	1	Comp6
14	Vcc	x	x	Vcc

#	4	4069 hex inverter		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1	1	2	Comp1
2	Out1	3	3	Icmp1
3	In2	1	1	Comp2
4	Out2	3	11	Icmp2
5	In3	1	13	Comp4
6	Out3	3	12	Icmp4
7	Gnd	x	x	Gnd
8	Out4	5	1	Icmp3
9	In4	1	14	Comp3
10	Out5	5	2	Icmp5
11	In5	2	2	Comp5
12	Out6	5	8	Icmp6
13	In6	2	1	Comp6
14	Vcc	x	x	Vcc

#	5	4073 tri 3 AND		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	4	8	Icmp3
2	In1b	4	10	Icmp5
3	In2a	x	x	Gnd
4	In2b	x	x	Gnd
5	In2c	x	x	Gnd
6	Out2	x	x	open
7	Gnd	x	x	Gnd
8	In1c	4	12	Icmp6
9	Out1	DB	8	W
10	Out3	x	x	open
11	In3a	x	x	Gnd
12	In3b	x	x	Gnd
13	In3c	x	x	Gnd
14	Vcc	x	x	Vcc

#	6	DATA BUS (DB)		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	VN	OUT3	x	VN
2	VE	OUT3	x	VE
3	VS	OUT3	x	VS
4	VW	OUT3	x	VW
5	N	OUT4	x	N
6	E	OUT4	x	E
7	S	OUT4	x	S
8	W	OUT4	x	W
9	col4	OUT5	x	col4
10	col2	OUT5	x	col2
11	col1	OUT5	x	col1
12	row4	OUT5	x	row4
13	row2	OUT5	x	row2
14	row1	OUT5	x	row1
15	Gnd	x	x	Gnd
16	Vcc	x	x	Vcc

#	7	RESISTOR Pack (RP)		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	R1in	OUT3	x	Vcc
2	R2in	OUT3	x	Vcc
3	R3in	OUT3	x	Vcc
4	R4in	OUT3	x	Vcc
5	R5in	OUT4	x	Vcc
6	R6in	OUT4	x	Vcc
7	R7in	m5#7	3	resist
8	R8in	m5#7	4	resist
9	R8out	m5#7	15	Nand1
10	R7out	m5#7	16	Nand2
11	R6out	2	1	Comp6
12	R5out	2	2	Comp5
13	R4out	1	13	Comp4
14	R3out	1	14	Comp3
15	R2out	1	1	Comp2
16	R1out	1	2	Comp1

Module #5 - MOVE CONTROL - Wiring Table B3

#	1	4053 tri 1 of 2 sw		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	B1	4	11	Arow2
2	B0	PC	13	Mrow2
3	C1	4	12	Arow4
4	C I/O	3	3	Nrow4
5	C0	PC	12	Mrow4
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	Csel	PC	1	A/Man
10	Bsel	same	9	A/Man
11	Asel	same	9	A/Man
12	A0	PC	14	Mrow1
13	A1	4	10	Arow1
14	A I/O	3	6	Nrow1
15	B I/O	3	4	Nrow2
16	Vcc	x	x	Vcc

#	2	4053 tri 1 of 2 sw		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	B1	5	11	Acol2
2	B0	PC	10	Mcol2
3	C1	5	12	Acol4
4	C I/O	3	11	Ncol4
5	C0	PC	9	Mcol4
6	INH	x	x	Gnd
7	Gnd	x	x	Gnd
8	Gnd	x	x	Gnd
9	Csel	PC	1	A/Man
10	Bsel	same	9	A/Man
11	Asel	same	9	A/Man
12	A0	PC	11	Mcol1
13	A1	5	10	Acol1
14	A I/O	3	14	Ncol1
15	B I/O	3	13	Ncol2
16	Vcc	x	x	Vcc

#	3	40174 hex D reg		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	CLR-	x	x	Vcc
2	Q1	4\DB	3\12	row4
3	D1	1	4	Nrow4
4	D2	1	15	Nrow2
5	Q2	4\DB	5\13	row2
6	D3	1	14	Nrow1
7	Q3	4\DB	7\14	row1
8	Gnd	x	x	Gnd
9	CLK	7	6	CYCLE
10	Q4	5\DB	3\9	col4
11	D4	2	4	Ncol4
12	Q5	5\DB	5\10	col2
13	D5	2	15	Ncol2
14	D6	2	14	Ncol1
15	Q6	5\DB	7\11	col1
16	Vcc	x	x	Vcc

#	4	4008 4 bit F Adder		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A4	x	x	Gnd
2	B3	DB	5	N
3	A3	3	2	row4
4	B2	same	2	N
5	A2	3	5	row2
6	B1	6	3	OR1
7	A1	3	7	row1
8	Gnd	x	x	Gnd
9	CI	x	x	Gnd
10	S1	1	13	Arow1
11	S2	1	1	Arow2
12	S3	1	3	Arow4
13	S4	x	x	open
14	CO	x	x	open
15	B4	same	2	N
16	Vcc	x	x	Vcc

#	5	4008 4 bit F Adder		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A4	x	x	Gnd
2	B3	DB	8	W
3	A3	3	10	col4
4	B2	same	2	W
5	A2	3	12	col2
6	B1	6	4	OR2
7	A1	3	15	col1
8	Gnd	x	x	Gnd
9	CI	x	x	Gnd
10	S1	2	13	Acol1
11	S2	2	1	Acol2
12	S3	2	3	Acol4
13	S4	x	x	open
14	CO	x	x	open
15	B4	same	2	W
16	Vcc	x	x	Vcc

#	6	4071 quad 2 in OR		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	DB	5	N
2	In1b	DB	7	S
3	Out1	4	6	OR1
4	Out2	5	6	OR2
5	In2a	DB	8	W
6	In2b	DB	6	E
7	Gnd	x	x	Gnd
8	In3a	x	x	Gnd
9	In3b	x	x	Gnd
10	Out3	x	x	open
11	Out4	x	x	open
12	In4a	x	x	Gnd
13	In4b	x	x	Gnd
14	Vcc	x	x	Vcc

#	7	4011 quad 2 in NAND		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	PC\RP	2\16	CyswL
2	In1b	same	1	CyswL
3	Out1	RP	7	resist
4	Out2	PR	8	resist
5	In2a	PC\RP	3\15	CyswH
6	In2b	same	5	CYCLE
7	Gnd	x	x	Gnd
8	In3a	x	x	Gnd
9	In3b	x	x	Gnd
10	Out3	x	x	open
11	Out4	x	x	open
12	In4a	x	x	Gnd
13	In4b	x	x	Gnd
14	Vcc	x	x	Vcc

#	8	PANEL CTRL BUS (PC		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	A/Man	PANout	x	A/Man
2	CyswL	PANout	x	CyswL
3	CyswH	PANout	x	CyswH
4	MmClr	PANout	x	MmClr
5	DSN	DB	5	DSN
6	DSE	DB	6	DSE
7	DSS	DB	7	DSS
8	DSW	DB	8	DSW
9	Mcol4	PANout	x	Mcol4
10	Mcol2	PANout	x	Mcol2
11	Mcol1	PANout	x	Mcol1
12	Mrow4	PANout	x	Mrow4
13	Mrow2	PANout	x	Mrow2
14	Mrow1	PANout	x	Mrow1
15	Gnd	x	x	Gnd
16	Vcc	x	x	Vcc

#	Mod4#6	DATA BUS (DB)		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	VN	OUT3	x	VN
2	VE	OUT3	x	VE
3	VS	OUT3	x	VS
4	VW	OUT3	x	VW
5	N	OUT4	x	N
6	E	OUT4	x	E
7	S	OUT4	x	S
8	W	OUT4	x	W
9	col4	OUT5	x	col4
10	col2	OUT5	x	col2
11	col1	OUT5	x	col1
12	row4	OUT5	x	row4
13	row2	OUT5	x	row2
14	row1	OUT5	x	row1
15	Gnd	x	x	Gnd
16	Vcc	x	x	Vcc

Module #6 - PATH OUTPUT DISPLAY - Wiring Table B4

#	12	DATA BUS (DB)		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	VN	OUT3	x	VN
2	VE	OUT3	x	VE
3	VS	OUT3	x	VS
4	VW	OUT3	x	VW
5	N	OUT4	x	N
6	E	OUT4	x	E
7	S	OUT4	x	S
8	W	OUT4	x	W
9	col4	OUT5	x	col4
10	col2	OUT5	x	col2
11	col1	OUT5	x	col1
12	row4	OUT5	x	row4
13	row2	OUT5	x	row2
14	row1	OUT5	x	row1
15	Gnd	x	x	Gnd
16	Vcc	x	x	Vcc

#	32	4069 hex inverter		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1	38	5	DScol1
2	Out1	33	1	ENcol1
3	In2	38	6	DScol2
4	Out2	33	8	ENcol2
5	In3	38	7	DScol3
6	Out3	34	1	ENcol3
7	Gnd	x	x	Gnd
8	Out4	34	8	ENcol4
9	In4	38	12	DScol4
10	Out5	35	1	ENcol5
11	In5	38	11	DScol5
12	Out6	35	8	ENcol6
13	In6	38	10	DScol6
14	Vcc	x	x	Vcc

#	33	4071 quad 2 in OR		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	32	2	ENcol1
2	In1b	37	1	row4
3	Out1	39	1	row4
4	Out2	39	15	Irow4
5	In2a	same	1	ENcol1
6	In2b	37	2	Irow4
7	Gnd	x	x	Gnd
8	In3a	32	4	ENcol2
9	In3b	same	2	row4
10	Out3	40	1	row4
11	Out4	40	15	Irow4
12	In4a	same	8	ENcol2
13	In4b	same	6	Irow4
14	Vcc	x	x	Vcc

#	34	4071 quad 2 in OR		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	32	6	ENcol3
2	In1b	37	1	row4
3	Out1	41	1	row4
4	Out2	41	15	Irow4
5	In2a	same	1	ENcol3
6	In2b	37	2	Irow4
7	Gnd	x	x	Gnd
8	In3a	32	8	ENcol4
9	In3b	same	2	row4
10	Out3	42	1	row4
11	Out4	42	15	Irow4
12	In4a	same	8	ENcol4
13	In4b	same	6	Irow4
14	Vcc	x	x	Vcc

#	35	4071 quad 2 in OR		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	32	10	ENcol5
2	In1b	37	1	row4
3	Out1	43	1	row4
4	Out2	43	15	Irow4
5	In2a	same	1	ENcol5
6	In2b	37	2	Irow4
7	Gnd	x	x	Gnd
8	In3a	32	12	ENcol6
9	In3b	same	2	row4
10	Out3	44	1	row4
11	Out4	44	15	Irow4
12	In4a	same	8	ENcol6
13	In4b	same	6	Irow4
14	Vcc	x	x	Vcc

#	36	4071 quad 2 in OR		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1a	37	12	ENcol7
2	In1b	37	1	row4
3	Out1	45	1	row4
4	Out2	45	15	Irow4
5	In2a	same	1	ENcol7
6	In2b	37	2	Irow4
7	Gnd	x	x	Gnd
8	In3a	x	x	Gnd
9	In3b	x	x	Gnd
10	Out3	x	x	open
11	Out4	x	x	open
12	In4a	x	x	Gnd
13	In4b	x	x	Gnd
14	Vcc	x	x	Vcc

#	37	4069 hex inverter		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	In1	DB	12	row4
2	Out1	33-36	6	Irow4
3	In2	DB	9	col4
4	Out2	38	15	Icol4
5	In3	x	x	Gnd
6	Out3	x	x	open
7	Gnd	x	x	Gnd
8	Out4	x	x	open
9	In4	x	x	Gnd
10	Out5	x	x	open
11	In5	x	x	Gnd
12	Out6	36	1	ENcol7
13	In6	38	9	DScol7
14	Vcc	x	x	Vcc

#	38	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	DB	9	col4
2	Asel1	DB	11	col1
3	Asel2	DB	10	col2
4	0	x	x	open
5	1	32	1	DScol1
6	2	32	3	DScol2
7	3	32	5	DScol3
8	Gnd	x	x	Gnd
9	7	37	13	DScol7
10	6	32	13	DScol6
11	5	32	11	DScol5
12	4	32	9	DScol4
13	Bsel2	same	3	col2
14	Bsel1	same	2	col1
15	Bdisab	37	4	Icol4
16	Vcc	x	x	Vcc

Module #6 - PATH OUTPUT DISPLAY - Wiring Table B4, continued

#	39	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	33	3	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	46	4	s1
6	2	46	6	s8
7	3	46	12	s15
8	Gnd	x	x	Gnd
9	7	47	12	s43
10	6	47	6	s36
11	5	47	4	s29
12	4	46	14	s22
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	33	4	Irow4
16	Vcc	x	x	Vcc

#	40	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	33	10	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	48	4	s2
6	2	48	6	s9
7	3	48	12	s16
8	Gnd	x	x	Gnd
9	7	49	12	s44
10	6	49	6	s37
11	5	49	4	s30
12	4	48	14	s23
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	33	11	Irow4
16	Vcc	x	x	Vcc

#	41	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	34	3	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	50	4	s3
6	2	50	6	s10
7	3	50	12	s17
8	Gnd	x	x	Gnd
9	7	51	12	s45
10	6	51	6	s38
11	5	51	4	s31
12	4	50	14	s24
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	34	4	Irow4
16	Vcc	x	x	Vcc

#	42	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	34	10	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	52	4	s4
6	2	52	6	s11
7	3	52	12	s18
8	Gnd	x	x	Gnd
9	7	53	12	s46
10	6	53	6	s39
11	5	53	4	s32
12	4	52	14	s25
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	34	11	Irow4
16	Vcc	x	x	Vcc

#	43	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	35	3	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	54	4	s5
6	2	54	6	s12
7	3	54	12	s19
8	Gnd	x	x	Gnd
9	7	55	12	s47
10	6	55	6	s40
11	5	55	4	s33
12	4	54	14	s26
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	35	4	Irow4
16	Vcc	x	x	Vcc

#	44	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	35	10	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	56	4	s6
6	2	56	6	s13
7	3	56	12	s20
8	Gnd	x	x	Gnd
9	7	57	12	s48
10	6	57	6	s41
11	5	57	4	s34
12	4	56	14	s27
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	35	11	Irow4
16	Vcc	x	x	Vcc

#	45	4555 dual 1of4 decd		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Adisab	36	3	row4
2	Asel1	DB	14	row1
3	Asel2	DB	13	row2
4	0	x	x	open
5	1	58	4	s7
6	2	58	6	s14
7	3	58	12	s21
8	Gnd	x	x	Gnd
9	7	59	12	s49
10	6	59	6	s42
11	5	59	4	s35
12	4	58	14	s23
13	Bsel2	same	3	row2
14	Bsel1	same	2	row1
15	Bdisab	36	4	Irow4
16	Vcc	x	x	Vcc

#	46	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	22	Path
2	Q1	OUT6	1	Path
3	R1	IN6	x	CLR
4	S1	39	5	s1
5	EN	x	x	Vcc
6	S2	39	6	s8
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	8	Path
10	Q3	OUT6	15	Path
11	R3	same	3	CLR
12	S3	39	7	s15
13	NC	x	x	open
14	S4	39	12	s22
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	47	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	29	Path
3	R1	IN6	x	CLR
4	S1	39	11	s29
5	EN	x	x	Vcc
6	S2	39	10	s36
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	36	Path
10	Q3	OUT6	43	Path
11	R3	same	3	CLR
12	S3	39	9	s43
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

Module #6 - PATH OUTPUT DISPLAY - Wiring Table B4, continued

#	48	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	23	Path
2	Q1	OUT6	2	Path
3	R1	IN6	x	CLR
4	S1	40	5	s2
5	EN	x	x	Vcc
6	S2	40	6	s9
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	9	Path
10	Q3	OUT6	16	Path
11	R3	same	3	CLR
12	S3	40	7	s16
13	NC	x	x	open
14	S4	40	12	s23
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	49	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	30	Path
3	R1	IN6	x	CLR
4	S1	40	11	s30
5	EN	x	x	Vcc
6	S2	40	10	s37
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	37	Path
10	Q3	OUT6	44	Path
11	R3	same	3	CLR
12	S3	40	9	s44
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	50	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	24	Path
2	Q1	OUT6	3	Path
3	R1	IN6	x	CLR
4	S1	41	5	s3
5	EN	x	x	Vcc
6	S2	41	6	s10
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	10	Path
10	Q3	OUT6	17	Path
11	R3	same	3	CLR
12	S3	41	7	s17
13	NC	x	x	open
14	S4	41	12	s24
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	51	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	31	Path
3	R1	IN6	x	CLR
4	S1	41	11	s31
5	EN	x	x	Vcc
6	S2	41	10	s38
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	38	Path
10	Q3	OUT6	45	Path
11	R3	same	3	CLR
12	S3	41	9	s45
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	52	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	25	Path
2	Q1	OUT6	4	Path
3	R1	IN6	x	CLR
4	S1	42	5	s4
5	EN	x	x	Vcc
6	S2	42	6	s11
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	11	Path
10	Q3	OUT6	18	Path
11	R3	same	3	CLR
12	S3	42	7	s18
13	NC	x	x	open
14	S4	42	12	s25
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	53	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	32	Path
3	R1	IN6	x	CLR
4	S1	42	11	s32
5	EN	x	x	Vcc
6	S2	42	10	s39
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	39	Path
10	Q3	OUT6	46	Path
11	R3	same	3	CLR
12	S3	42	9	s46
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	54	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	26	Path
2	Q1	OUT6	5	Path
3	R1	IN6	x	CLR
4	S1	43	5	s5
5	EN	x	x	Vcc
6	S2	43	6	s12
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	12	Path
10	Q3	OUT6	19	Path
11	R3	same	3	CLR
12	S3	43	7	s19
13	NC	x	x	open
14	S4	43	12	s26
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	55	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	33	Path
3	R1	IN6	x	CLR
4	S1	43	11	s33
5	EN	x	x	Vcc
6	S2	43	10	s40
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	40	Path
10	Q3	OUT6	47	Path
11	R3	same	3	CLR
12	S3	43	9	s47
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	56	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	27	Path
2	Q1	OUT6	6	Path
3	R1	IN6	x	CLR
4	S1	44	5	s6
5	EN	x	x	Vcc
6	S2	44	6	s13
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	13	Path
10	Q3	OUT6	20	Path
11	R3	same	3	CLR
12	S3	44	7	s20
13	NC	x	x	open
14	S4	44	12	s27
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

Module #6 - PATH OUTPUT DISPLAY - Wiring Table B4, continued

#	57	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	34	Path
3	R1	IN6	x	CLR
4	S1	44	11	s34
5	EN	x	x	Vcc
6	S2	44	10	s41
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	41	Path
10	Q3	OUT6	48	Path
11	R3	same	3	CLR
12	S3	44	9	s48
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	58	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	OUT6	28	Path
2	Q1	OUT6	7	Path
3	R1	IN6	x	CLR
4	S1	45	5	s7
5	EN	x	x	Vcc
6	S2	45	6	s14
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	14	Path
10	Q3	OUT6	21	Path
11	R3	same	3	CLR
12	S3	45	7	s21
13	NC	x	x	open
14	S4	45	12	s28
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

#	59	4043 quad R/S ff		
Chip		Connection		
Pin	Desc	Chip	Pin	Desc
1	Q4	x	x	open
2	Q1	OUT6	35	Path
3	R1	IN6	x	CLR
4	S1	45	11	s35
5	EN	x	x	Vcc
6	S2	45	10	s42
7	R2	same	3	CLR
8	Gnd	x	x	Gnd
9	Q2	OUT6	42	Path
10	Q3	OUT6	49	Path
11	R3	same	3	CLR
12	S3	45	9	s49
13	NC	x	x	open
14	S4	x	x	Gnd
15	R4	same	3	CLR
16	Vcc	x	x	Vcc

Appendix C

Maze Machine Tools/ Parts Required

The electronic 'Maze Machine' required funds in the vicinity of $360 to purchase the needed off-the-shelf IC chips and accessory components necessary to build the system. Thesis director, Dr. John B. Cheatham, Jr., provided monetary support for building the machine through a contract and grant from NASA/ JSC and RICIS.

The following table shows the parts inventory for the completed Maze Machine. Note that 65 IC chips are used to construct the machine, however 28 of the chips are simply used to store the path output display by way of RS flip-flop chips for memory and an array of LEDs to highlight the path.

As constructed, the machine requires an 'awkward' system of hand-placed plugs to set the maze environment. This method was employed due to low cost. An alternative system could use CMOS 4066 (quad digital/analog bilateral switchs), 4503 (tri-state hex buffers), and 4043 (quad R/S flip-flop) chips, (thus integrating Modules 1 and 2). This was not pursued since the machine was only built to prove a concept and any practical sized machine would require custom analog VLSI chips.

A practical implementation of this technology would be through the use of a custom add-in board for a microcomputer, which would use the aforementioned VLSI chips. The computer would be used for interfacing with both a semiautonomous mobile robot (and handle the robot's numerous other functions) and the 'Maze Machine' on-a-board. The board would perform its path planning functions (in real time, due to the parallel processing nature of the connectionist network) while the computer directs the robot's actions.

MAZE MACHINE, Electronic Hybrid Network Parts Inventory

IC Chip#	Item Description	Cost (ea)	Module Requirements 1	2	3	4	5	6	Qty rqd	Cost total
339	quad comparator	$0.55				2			2	$1.10
4008	4 bit full adder (or 74C83)	$1.50			4		2		6	$9.00
4011	quad 2 input NAND	$0.30					1		1	$0.30
4043	quad RS ff (NOR logic)	$0.20						14	14	$2.80
4051	1 of 8 analog switch	$1.50			19				19	$28.50
4053	triple SPDT chip	$1.50					2		2	$3.00
4069	hex inverter	$0.55				1		2	3	$1.65
4071	quad 2 input OR	$0.30					1	4	5	$1.50
4073	3 input AND	$0.30				2			2	$0.60
4555	dual 1 of 4 decoder non-inv	$0.99						8	8	$7.92
40174	hex-D storage register	$1.50					1		1	$1.50
74C48	BCD to 7 seg decoder	$1.50					2		2	$3.00
x	510Ω resistor 1/4w	$0.03	100				4		104	$3.12
x	10kΩ resistor 1/4w	$0.03				6			6	$0.18
x	100kΩ resistor 1/4w	$0.03		84			2		86	$2.58
x	8 pin IC socket, wire wrap	$0.40	100	49					149	$59.60
x	14 pin IC socket, wire wrap	$0.75				5			5	$3.75
x	16 pin IC socket, wire wrap	$0.75			24	1	7	24	56	$42.00
x	LED flashing , green, GOAL	$0.85	1						1	$0.85
x	LED green, round, 5mm PATH	$0.15	49						49	$7.35
x	LED red, round, 5mm OBST	$0.15	49				4		53	$7.95
x	LED yellow, rnd, 5mm START	$0.15	49						49	$7.35
x	LED sockets, 5mm	$0.10					4		4	$0.40
x	decimal LED display	$1.79					2		2	$3.58
x	ribbon connection, 16 strand	$3.00			1		1		2	$6.00
x	ribbon connection, 50 strand	$14.00		2					2	$28.00
x	switch SPDT	$2.89					7		7	$20.23
x	switch SPDT, momentary	$3.69					1	1	2	$7.38
x	protoboard 8" x 6"	$6.00		1	1	1			3	$18.00
x	wire, wire wrap, 50ft	$5.20	4						4	$20.80
x	Socket, banana plug	$1.50						2	2	$3.00
x	power supply, 12v DC	$30.00						1	1	$30.00
x	cabinet, alum, 13x10x4 in.	$22.94						1	1	$22.94
									Grand Total	$355.93

Table C1

Maze Machine Parts Inventory

Appendix D

Procedure for Running Maze Machine Auto-Path Test

The following steps refer to switches on the front face of the Maze Machine cabinet. (Figure D1 is a diagram of the front of the Maze Machine.)

1. Establish/ define the maze environment by identifying obstacles, free spaces, the start node, and the goal node. Place the required plugs in the appropriate node sockets on the top of the Maze Machine. (Ensure that the plugs are oriented in the right direction.) The plugs can be easily identified in the following manner: obstacle plugs have red stickers, free spaces have green LEDs and green stickers, the start has a green LED and a yellow sticker, and the goal has a regular green path LED plus a blinking green LED.

2. With the machine in the MANUAL mode, use the "manual current node input" switches to set the start point (i.e. row and column, using binary representation). Push the cycle button to load the data to the Move Control Module. The start node LED will light and the current node display will indicate the current location digitally (i.e. row and column). Also, one red direction indicator LED will light, indicating the direction in which to move.

3. Next, place the machine in the AUTO mode. Push the cycle button to take one step along the path. The appropriate path LED on the top of the machine will light and the current node location digital display and direction indicator LED will become updated accordingly. Continue to take one step at at time, by pushing the cycle button, until the entire path is lit-up and the goal node is reached.

4. To start over and clear the maze path display, simply push the spring loaded MEM/ CLR toggle switch to CLR and release. The machine will now be reset and all maze LEDs will be off except the blinking goal.

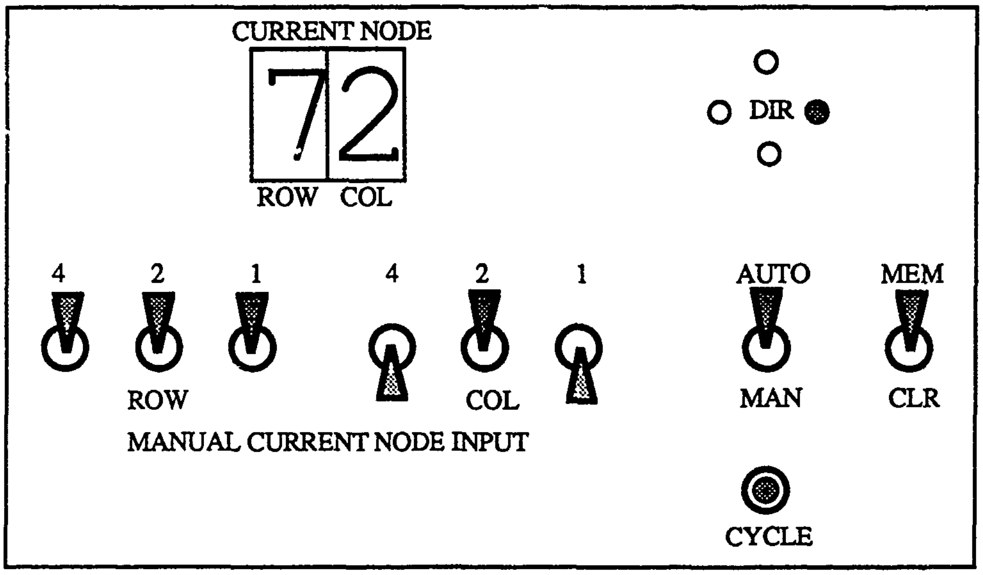


Figure D1

Maze Machine External Input and Output Display

NOTE:

To read out the current node voltages, plug a voltmeter into the back of the machine at the marked socket and either take readings as you step through a maze problem's path or use the MANUAL switch and manual current node input toggles to select the desired node(s) for voltage sampling.

Appendix E

Maze Machine Test Data and Results

The Maze Machine was tested using a variety of sample 'mazes' / path planning and obstacle avoidance problems. The machine consistently provided good (near-optimal) solutions based on the environment and assumptions imposed.

In the tests ran, the environment usually contained more than one feasible path (from start to goal). After setting the environment (by defining start, goal, free space, and obstacle nodes), voltage readings were taken for each node in the 'maze'. After sampling the voltage outputs, the test navigation problem was run on the Maze Machine. As stated earlier, in each case, the machine successfully navigated the maze and avoided obstacles while selecting a good (best, from a local voltage increase basis) path. The results of some sample tests that were run are included as tables in this appendix. Their significance are as follows:

Table E1: Test 1 shows the maze from Figure 2 in the Introduction Chapter. First the 'Maze Mask' with start and goal positions is displayed. Next, the Vout Sampling Table with voltage potential output values listed per node. Finally the Auto-Path Results are shown

T [illegible] t 2a shows a new maze layout, with 'Maze Mask' + start & goal, Vout Samp [illegible] nd Auto-Path Results.

Table E3: Test 2b uses the same 'Maze Mask' as Test 2a, however the start position has been altered. Note new Vout Sampling Table and Auto-Path Results.

Table E4: Test 3a shows a new maze layout where the 'Maze Mask' contains no obstacles. This test was run to show the tolerance errors in the hardware system. Note

that since the start & goal were placed in opposing corners the system is symmetrical. The Vout readings should be identical on the cross diagonals (perpendicular to straight line between start and goal). The small errors found in the actual Vout readings can be accounted for easily by the tolerances in the components that make up the Maze Machine (especially the resistors of Module 2, each with a 5% manufactures' tolerance). Also note that any path that moves only right or down (East or South) is equally good based on the assumptions/ limitations made.

Table E5: Test 3b uses the same no obstacle 'Maze Mask' as Test 3a, however the start position has been altered. Note new Vout Sampling Table and Auto-Path Results.

Table E6: Test 4 shows the maze from Figure 14 and Table 1 in Chapter 4. Here again, the 'Maze Mask' with start and goal positions, along with the Vout Sampling Table and Auto-Path Results are shown.

MAZE MACHINE, Test Run Printout **Test 1**

Maze Mask + External Inputs:

	1	2	3	4	5	6	7
1	Start	XXX					XXX
2		XXX		XXX	XXX		
3				XXX	XXX	XXX	
4		XXX		XXX			
5		XXX	XXX	XXX		XXX	
6			XXX			XXX	
7	XXX					XXX	Goal

Vout Sampling:

	1	2	3	4	5	6	7
1	0.00	0.00	4.43	4.98	5.53	6.10	0.00
2	1.11	0.00	3.88	0.00	0.00	6.68	7.25
3	2.22	2.77	3.35	0.00	0.00	0.00	7.84
4	2.72	0.00	3.28	0.00	7.35	7.89	8.47
5	3.24	0.00	0.00	0.00	6.80	0.00	9.57
6	3.75	4.25	0.00	6.02	6.32	0.00	10.73
7	0.00	4.75	5.25	5.80	6.01	0.00	11.92

Auto Path Results:

	1	2	3	4	5	6	7
1	v	XXX	>	>	>	v	XXX
2	v	XXX	^	XXX	XXX	>	v
3	>	>	^	XXX	XXX	XXX	v
4		XXX		XXX			v
5		XXX	XXX	XXX		XXX	v
6			XXX			XXX	v
7	XXX					XXX	Goal

Comments:

Successfully navigated the maze while minimizing the number of moves required.

Table E1

MAZE MACHINE, Test Run Printout **Test 2a**

Maze Mask + External Inputs:

	1	2	3	4	5	6	7
1		XXX					Goal
2		XXX				XXX	
3				XXX		XXX	
4		XXX	XXX	XXX		XXX	
5		Start		XXX		XXX	
6				XXX			
7		XXX					

Vout Sampling:

	1	2	3	4	5	6	7
1	3.30	0.00	7.30	7.82	8.66	10.28	11.95
2	3.25	0.00	6.83	7.51	7.88	0.00	10.89
3	3.38	4.52	5.66	0.00	7.43	0.00	9.88
4	2.21	0.00	0.00	0.00	7.01	0.00	8.91
5	1.05	0.00	0.73	0.00	6.60	0.00	7.95
6	0.92	0.79	1.44	0.00	6.22	6.49	7.02
7	0.91	0.00	2.81	4.17	5.54	6.21	6.59

Auto Path Results:

	1	2	3	4	5	6	7
1		XXX			>	>	Goal
2		XXX	>	>	^	XXX	
3	>	>	^	XXX		XXX	
4	^	XXX	XXX	XXX		XXX	
5	^	<		XXX		XXX	
6				XXX			
7		XXX					

Comments:

Successfully navigated the maze while minimizing the number of moves required.

Table E2

MAZE MACHINE, Test Run Printout Test 2b

Maze Mask + External Inputs:

	1	2	3	4	5	6	7
1		XXX					Goal
2		XXX				XXX	
3				XXX		XXX	
4		XXX	XXX	XXX		XXX	
5				XXX		XXX	
6				XXX			
7	Start	XXX					

Vout Sampling:

	1	2	3	4	5	6	7
1	4.86	0.00	8.17	8.62	9.29	10.60	11.95
2	4.92	0.00	7.80	8.36	8.66	0.00	11.10
3	4.97	5.89	6.83	0.00	8.31	0.00	10.28
4	4.02	0.00	0.00	0.00	8.00	0.00	9.52
5	3.08	3.18	3.44	0.00	7.69	0.00	8.76
6	2.03	2.98	3.75	0.00	7.41	7.62	8.04
7	0.00	0.00	4.78	5.82	6.89	7.40	7.68

Auto Path Results:

	1	2	3	4	5	6	7
1		XXX			>	>	Goal
2		XXX	>	>	^	XXX	
3	>	>	^	XXX		XXX	
4	^	XXX	XXX	XXX		XXX	
5	^			XXX		XXX	
6	^			XXX			
7	^	XXX					

Comments:

Successfully navigated the maze while minimizing the number of moves required.

Table E3

MAZE MACHINE, Test Run Printout Test 3a

Maze Mask + External Inputs:

	1	2	3	4	5	6	7
1	Start						
2							
3							
4							
5							
6							
7							Goal

Vout Sampling:

	1	2	3	4	5	6	7
1	0.00	2.32	3.71	4.63	5.27	5.69	5.87
2	2.31	3.25	4.16	4.93	5.51	5.91	6.10
3	3.70	4.16	4.77	5.38	5.91	6.32	6.52
4	4.63	4.93	5.39	5.93	6.45	6.91	7.18
5	5.27	5.41	5.91	6.46	7.06	7.67	8.11
6	5.[illegible]	5.91	6.31	6.92	7.68	8.60	9.51
7	5.87	6.10	6.52	7.17	8.10	9.50	11.84

Auto Path Results:

	1	2	3	4	5	6	7
1	>	>	>	>	>	v	
2						v	
3						v	
4						v	
5						v	
6						>	v
7							Goal

Comments:

Successfully navigated the 'non-maze' while minimizing the number of moves required.

Table E4

MAZE MACHINE, Test Run Printout Test 3b

Maze Mask + External Inputs:

	1	2	3	4	5	6	7
1							Goal
2							
3							
4							
5							
6							
7	Start						

Vout Sampling:

	1	2	3	4	5	6	7
1	6.15	6.21	6.55	7.20	8.11	9.52	11.84
2	5.52	5.89	6.30	6.91	7.66	8.57	9.48
3	5.17	5.45	5.87	6.42	7.01	7.63	8.05
4	4.58	4.89	5.34	5.90	6.42	6.88	7.14
5	3.66	4.12	4.72	5.35	5.86	6.26	6.46
6	2.29	3.21	4.12	4.89	5.46	5.86	6.04
7	0.00	2.30	3.65	4.56	5.20	5.62	5.80

Auto Path Results:

	1	2	3	4	5	6	7
1						>	Goal
2						^	
3						^	
4						^	
5						^	
6						^	
7	>	>	>	>	>	^	

Comments:

Successfully navigated the 'non-maze' while minimizing the number of moves required.

Table E5

MAZE MACHINE, Test Run Printout Test 4

Maze Mask + External Inputs:

	1	2	3	4	5	6	7
1	XXX						Goal
2			XXX		XXX	XXX	XXX
3		XXX					XXX
4	XXX	XXX	XXX		XXX		
5					XXX	XXX	
6		XXX	XXX		XXX		
7		Start					

Vout Sampling:

	1	2	3	4	5	6	7
1	0.00	6.86	6.86	6.86	8.03	9.29	10.53
2	6.72	6.76	0.00	5.60	0.00	0.00	0.00
3	6.62	0.00	4.44	4.45	4.07	3.80	0.00
4	0.00	0.00	0.00	3.47	0.00	3.51	3.20
5	1.33	1.72	2.13	2.56	0.00	0.00	2.91
6	0.87	0.00	0.00	2.06	0.00	2.40	2.70
7	0.44	0.00	0.80	1.60	1.83	2.12	2.09

Auto Path Results:

	1	2	3	4	5	6	7
1	XXX			>	>	>	Goal
2			XXX	^	XXX	XXX	XXX
3		XXX		^			XXX
4	XXX	XXX	XXX	^	XXX		
5				^	XXX	XXX	
6		XXX	XXX	^	XXX		
7		>	>	^			

Comments:

Successfully navigated the maze while minimizing the number of moves required.

Table E6

Appendix F

AMAZ3D.f Source Code Listing

The AMAZ3D.f program code was written in FORTRAN77. The main program AMAZ3D.f along with its subroutines comprise the complete serial computer simulation for the hybrid connectionist network system used for path planning and obstacle avoidance. The subprograms used are listed below, along with short explanations of there functions:

program AMAZ3D is the main program for finding good paths through 3D mazes.

subroutine MAZLGO(UNIT) prints the AMAZ3D logo to device labeled U.

subroutine MAZINP inputs maze data and set WTS array.

subroutine MAZOUT(U) outputs maze environment (and any path data such as start, goal and path steps if calculated) to device labeled U.

subroutine PAROUT(U) outputs parameter/ maze data to device labeled U.

subroutine UPDATE allows user to modify maze/ parameter data before going to iteration process.

subroutine MAZPOT converts maze input data from a character array MAZ to a value array POT, plus it initializes the array PFLD (the nodal voltage potentials array).

subroutine MAZFLD calculates final values for potential field array by iteration process and shows in-progress error calculations for every 10 iterations.

subroutine MAZFNL(U) outputs nodal voltage potential field to device labeled U.

subroutine MAZMOV calculates path in a step-by-step procedure using locally optimal moves/ steps.

subroutine MAZPTH(U) outputs the solution path in a list format (includes path node vector addresses and next move directions) to device labeled U.

function MAZDIR(IDIR) produces direction character string from integer input (i.e. N, E, S, W,NE, SE, SW, and NW).

subroutine bell(NUM) rings bell NUM times to alert user to input requirements.

The AMAZ3D program also reads in three data files:

amaz3d.prf (A preferences file for: 1) describing the screen output device number, 6 for IBM PC and SUN workstations, 9 for Macintosh PC and 2) FILEDF, the default file-name for the assumed input parameter file.),

FILEDF or file-name entered by user (A parameter file which provides key information to the program. All parameters but the sensory data file array dimensions and FILEM, the initial node values file, can be modified later by the user within the AMAZ3D program.),

FILEM (Sensory data input file, name provided by above mentioned parameter file, which provides AMAZ3D program with initial environmental sensory data for the nodal array which was dimensioned through FILEDF entries.).

Notes:

1) To keep the program memory requirements reasonable for a microcomputer, the maximum sensory data input file has been limited to a three dimensional array of 80 rows by 80 columns by 8 layers of height (thereby requiring approximately 1 Mbyte RAM). These values are arbitrary, of course, and can be changed in the variable declaration statements of the program source code before compilation. The only limitation is memory availability.

2) Also note that two types of obstacles have been allowed for: 1) 'normal' obstacles are set to GND and are isolated from their neighbors in the network (i.e. do not influence the neighbors' nodal value outputs) and 2) 'connected' obstacles which have the same characteristics as the start node (i.e. set to GND but left connected to the network so that their influence is felt by the neighboring free nodes). This second type of obstacle node allows the program to approximate the potential fields approach used by Norwood (1989) in his Master's Thesis on "Robotic Path Planning and Obstacle Avoidance: A Neural Network Approach".

Program AMAZ3D.f Source Code Listing

```
c----+---1---------2---------3---------4---------5---------6---------7--

      program AMAZ3D
c  main program for finding good paths through 3D mazes
c declaration of global/common variables
      integer        SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
      integer        MAXIT,CNT,MOVE,I,J,K,DIRALL
      real           OBSX,OBSC,GND,VCC,DIST
      real           ALLERR,ACTERR,TTLERR
      integer        WTS(0:81,0:81,0:9)
      real           POT(0:81,0:81,0:9)
      real           PFLD(0:81,0:81,0:9)
      integer        PATH(0:1000,4)
      character*12   FILEDF,FILEP,FILEM,FILEO
      character*1    MAZ(0:81,0:81,0:9)
      character*1    OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
      common         SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2                CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                ALLERR,ACTERR,TTLERP,WTS,
     4                POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c declare local variables
      character*1    BYPASS,QUITER
      logical        PRFEXS
c  SCR = screen device/unit # default
c  FILEDF = default parameter input file
c read in preference file for SCR,FILEDF
      inquire(file='amaz3d.prf',exist=PRFEXS)
      if (.not.PRFEXS) then
        pause 'Needed preference file "amaz3d.prf" not found! '
        stop
      endif
      open(2,file='amaz3d.prf')
      read(2,2000) SCR,FILEDF
2000  format(i1/,a12/)
      close(2)
c  FIL = output-file unit default
      FIL  = 8
c print logo to screen
10    call MAZLGO(SCR)
c  input parameter data
      call PARINP
c  input maze data
      call MAZINP
c  update maze/parameter data
      write(*,*) 'Maze Environment... '
      call MAZOUT(SCR)
      if (SCR.eq.9) call bell(1)
      if (SCR.eq.9) pause 'press <CR> to continue... '
      if (SCR.eq.9) write(*,*)
      call PAROUT(SCR)
      call UPDATE
c  output maze/parameter data to file
      write(FIL,*) 'Maze Environment... '
      call MAZOUT(FIL)
      call PAROUT(FIL)
c  create real valued array from char array
      call MAZPOT
c  calc final values for pot fld array
```

```
      call MAZFLD
      call bell(1)
      write(*,2010)
2010  format('Enter <Y> to printout the pot. values array, or',/
     +        '      <CR> to bypass output of potential values : '$)
      BYPASS=' '
      read(*,'(a1)') BYPASS
      write(*,*)
      if (BYPASS.ne.'Y' .and. BYPASS.ne.'y') goto 20
c  output final potential field
      call MAZFNL(SCR)
      call MAZFNL(FIL)
c  output final iteration info
20    write(*,2030)   CNT,ACTERR,TTLERR
      write(FIL,2030) CNT,ACTERR,TTLERR
2030  format(/'                Total Iterations  = ',i8,/
     2        'Maximum individual nodal change  = ',f8.4,/
     3        '   Total iteration nodal change  = ',f8.4//)
      if (SCR.eq.9) call bell(1)
      if (SCR.eq.9) pause 'press <CR> to continue... '
      if (SCR.eq.9) write(*,*)
c  calculate path
      call MAZMOV
c  output path
      call MAZPTH(SCR)
      call MAZPTH(FIL)
      if (SCR.eq.9) call bell(1)
      if (SCR.eq.9) pause 'press <CR> to continue... '
      if (SCR.eq.9) write(*,*)
      write(*,*)
c  output maze solution (in character form)
      write(*,*) 'Maze Solution... '
      call MAZOUT(SCR)
      write(FIL,*)
      write(FIL,*) 'Maze Solution... '
      call MAZOUT(FIL)
      write(*,*)
      write(*,*) 'NOTE:  Duplicate A-MAZ3D output',
     +           ' written to file ',FILEO
      close(FIL)
      call bell(1)
      write(*,2040)
2040  format (/'Enter <Y> to restart program, or',/
     +         '      <CR> to quit : '$)
      QUITER=' '
      read(*,'(a1)') QUITER
      write(*,*)
      if (QUITER.eq.'Y' .or. QUITER.eq.'y') goto 10
      end
```

```
c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine MAZLGO(UNIT)
c  printout logo
c  declare local variable
       integer UNIT
       write(UNIT,2000)
2000   format(/
     2  '%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%',/
     3  '%                                                       %',/
     4  '%      A      M     M       A      ZZZZZ    333  DDDD   %',/
     5  '%     A A     MM   MM      A A        Z       3  D   D  %',/
     6  '%    AAAAA    M M M M     AAAAA      Z       33  D   D  %',/
     7  '%   A     A   M  M  M    A     A    Z         3  D   D  %',/
     8  '%  A       A  M     M   A       A  ZZZZZ    333  DDDD   %',/
     9  '%                                                       %',/
     +  '%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%',/
     1  'Chris Schuster    MEMS/Robotics,Rice University     v6.03',/
     2  /)
       return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine PARINP
c  input parameter/maze data
c declaration of global/common variables
       integer        SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
       integer        MAXIT,CNT,MOVE,I,J,K,DIRALL
       real           OBSX,OBSC,GND,VCC,DIST
       real           ALLERR,ACTERR,TTLERR
       integer        WTS(0:81,0:81,0:9)
       real           POT(0:81,0:81,0:9)
       real           PFLD(0:81,0:81,0:9)
       integer        PATH(0:1000,4)
       character*12   FILEDF,FILEP,FILEM,FILEO
       character*1    MAZ(0:81,0:81,0:9)
       character*1    OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
       common         SCR,FIL,ROW,COL,DEP,STARTN,COALN,MAXIT,
     2                 CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                 ALLERR,ACTERR,TTLERR,WTS,
     4                 POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                 MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c declare local variables
       logical        PAREXS
c  get parameter file-name, (default parameter_file = FILEDF)
10     FILEP=FILEDF
       call bell(1)
       write(*,2000) FILEP
2000   format('Enter <CR> for the default parameter file ',a12,/
     +        '   or <file_name> of a custom parameter file : '$)
       read(*,'(a12)') FILEP
       if (FILEP.eq.'            ') FILEP=FILEDF
       write(*,*)
       write(*,*)
       inquire(file=FILEP,exist=PAREXS)
       if (PAREXS) goto 20
       write(*,*) '       Parameter file ',FILEP,' not found! '
       write(*,*)
       goto 10
c  input program parameters
20     FILEDF = FILEP
```

```
      open(1,file=FILEP)
      read(1,2010) ROW,COL,DEP,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,
     2              GOAL,STARTN(1),STARTN(2),STARTN(3),GOALN(1),
     3              GOALN(2),GOALN(3),OBSX,OBSC,
     4              GND,VCC,ALLERR,MAXIT,FILEM,FILEO,DIRALL
2010  format (3(i3/),7(a1/),6(i3/),5(g12.5/),i5/,2(a12/),i1/)
      close(1)
      inquire(file=FILEM,exist=PAREXS)
      if (PAREXS) goto 8888
      write(*,*) '    Maze data file ',FILEM,' not found!'
      write(*,*)
      goto 10
8888  return
      end

c----+---1---------2---------3---------4---------5---------6---------7--

      subroutine MAZINP
c  input maze data, set WTS array
c declaration of global/common variables
      integer       SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
      integer       MAXIT,CNT,MOVE,I,J,K,DIRALL
      real          OBSX,OBSC,GND,VCC,DIST
      real          ALLERR,ACTERR,TTLERR
      integer       WTS(0:81,0:81,0:9)
      real          POT(0:81,0:81,0:9)
      real          PFLD(0:81,0:81,0:9)
      integer       PATH(0:1000,4)
      character*12  FILEDF,FILEP,FILEM,FILEO
      character*1   MAZ(0:81,0:81,0:9)
      character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
      common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2               CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3               ALLERR,ACTERR,TTLERR,WTS,
     4               POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5               MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
      integer       INK
c  initialize maze outer boundaries to obstacle nodes
c  init top and bottom of maze
      do 20 J=0,COL+1
        do 10 I=0,ROW+1
          MAZ(I,J,0)=OBSTX
          MAZ(I,J,DEP+1)=OBSTX
10      continue
20    continue
c  init sides of maze
      do 50 K=1,DEP
        do 30 J=0,COL+1
          MAZ(0,J,K)=OBSTX
          MAZ(ROW+1,J,K)=OBSTX
30      continue
        do 40 I=1,ROW
          MAZ(I,0,K)=OBSTX
          MAZ(I,COL+1,K)=OBSTX
40      continue
50    continue
c  read in maze data (layer by layer)
      open(1,file=FILEM)
      do 70 K=1,DEP
        if (DEP.ne.1) then
          read(1,'(i3)') INK
          if (INK.ne.K) then
            pause 'error reading in maze data file !!!'
            stop
          endif
        endif
        do 60 I=1,ROW
          read(1,2000) (MAZ(I,J,K),J=1,COL)
2000      format(128a1)
60      continue
70    continue
      close(1)

      return
      end

c----+---1---------2---------3---------4---------5---------6---------7--

      subroutine MAZOUT(U)
c     output extended maze to screen and file FILEO
c declaration of global/common variables

      integer       SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
      integer       MAXIT,CNT,MOVE,I,J,K,DIRALL
      real          OBSX,OBSC,GND,VCC,DIST
      real          ALLERR,ACTERR,TTLERR
      integer       WTS(0:81,0:81,0:9)
      real          POT(0:81,0:81,0:9)
      real          PFLD(0:81,0:81,0:9)
      integer       PATH(0:1000,4)
      character*12  FILEDF,FILEP,FILEM,FILEO
      character*1   MAZ(0:81,0:81,0:9)
      character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
      common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2               CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3               ALLERR,ACTERR,TTLERR,WTS,
     4               POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5               MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
      integer       COLUMN(0:129)
      integer       U
c  output initial maze parameters
      write(U,2000) FILEM,ROW,COL,DEP,OBSTX,OBSTC,FREE1,FREE2,FREE3,
     +               START,(STARTN(I),I=1,3),GOAL,(GOALN(I),I=1,3)
2000  format(/,' The Maze:   ',a12,' has ',i3,' Rows ,',
     2   i3,' Cols ,',i3,' Depth Layer(s)',
     3   //' Key: Obstacles  = ',a1,' (isolated) ,   ',a1,
     4   ' (connected)',/'        Free Space = ',a1,3x,
     5   '[ + ',a1,' (doors) ,   ',a1,' (elevators) ]',/
     7   '         Start node = ',a1,' , at (',i3,',',i3,',',i3,')',/
     8   '         Goal  node = ',a1,' , at (',i3,',',i3,',',i3,')',/)
c  create column header
      do 20 I=0,12
        do 10 J=0,9
          COLUMN(10*I+J)=J
10      continue
20    continue
c  output maze
c  first check if DEP = 1 or
c  horizontal printout of layers is practical
      if (DEP.eq.1 .or. (COL+3)*DEP .gt. 80) goto 100
c  for horz layers:  output depth layer titles
        write(U,2010)
```

```
2010      format('    ',$)
       do 30, K=1,DEP
         write(U,2020) K
2020      format('   Depth',i2,$)
30     continue
       write(U,*)
c  output column headers
       write(U,2030)((char(48+COLUMN(J)),J=1,COL),' ',' ',' ',K=1,DEP)
2030   format('      ',128a1)
       write(U,*)
c  output row headers and mazes
         do 40, I=1,ROW
           write(U,2040) I,((MAZ(I,J,K),J=1,COL),' ',' ',' ',K=1,DEP)
2040       format(i3,3x,128a1)
40       continue
       goto 8888
c  for layers printed out vertically:
c  output column header
100    write(U,2050)  (COLUMN(J),J=1,COL)
2050   format('      ',128i1)
       write(U,*)
       do 120, K=1,DEP
         if (DEP.ne.1) then
           write(U,*)
           write(U,*) 'Depth Layer:',K
         endif
         do 110, I=1,ROW
           write(U,2060) I,(MAZ(I,J,K),J=1,COL)
2060       format(i3,3x,128a1)
110      continue
120    continue
8888   return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine PAROUT(U)
c  output parameter/maze data
c declaration of global/common variables
       integer        SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
       integer        MAXIT,CNT,MOVE,I,J,K,DIRALL
       real           OBSX,OBSC,GND,VCC,DIST
       real           ALLERR,ACTERR,TTLERR
       integer        WTS(0:81,0:81,0:9)
       real           POT(0:81,0:81,0:9)
       real           PFLD(0:81,0:81,0:9)
       integer        PATH(0:1000,4)
       character*12   FILEDF,FILEP,FILEM,FILEO
       character*1    MAZ(0:81,0:81,0:9)
       character*1    OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
       common         SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2                 CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                 ALLERR,ACTERR,TTLERR,WTS,
     4                 POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                 MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
       integer        U
c  write node potential parameters
       write(U,2000)  OBSX,OBSC,GND,VCC,ALLERR,MAXIT,DIRALL
2000   format (/'Node Potential Parameters:',//
     2        '  Isolated  obstacle value  (ObsX)  = ',f8.4,/
     3        '  Connected obstacle value  (ObsC)  = ',f8.4,/
     4          '  Start node value  - - - -  (Gnd)  = ',f8.4,/
     5          '  Goal  node value  - - - -  (Vcc)  = ',f8.4,/
     6          '  Max nodal change allowed per iter = ',f8.4,/
     7          '  Max number of iterations allowed  = ',i8,/
     8          '  Horiz directions allowed (4 or 8) = ',i8/)
       return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine UPDATE
c  allows user to modify maze/parameter data
c declaration of global/common variables
       integer       SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
       integer       MAXIT,CNT,MOVE,I,J,K,DIRALL
       real          OBSX,OBSC,GND,VCC,DIST
       real          ALLERR,ACTERR,TTLERR
       integer       WTS(0:81,0:81,0:9)
       real          POT(0:81,0:81,0:9)
       real          PFLD(0:81,0:81,0:9)
       integer       PATH(0:1000,4)
       character*12  FILEDF,FILEP,FILEM,FILEO
       character*1   MAZ(0:81,0:81,0:9)
       character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
       common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2                CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                ALLERR,ACTERR,TTLERR,WTS,
     4                POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
       character*1   UPD,MOD,NCHAR
       integer       NROW,NCOL,NDEP
       UPD='N'
c  ask if modification is desired
       call bell(1)
       write(*,2000)
2000   format('Enter <Y>  to modify the parameters, or',/
     +         '       <CR> to use default parameters : '$)
       read(*,'(a1)') UPD
       write(*,*)
       if (UPD.ne.'Y' .and. UPD.ne.'y') goto 8000
       write(*,*)
1000   MOD='Q'
       write(*,2010)  (STARTN(I),I=1,3),(GOALN(I),I=1,3),
     +                 OBSX,OBSC,GND,VCC,ALLERR,MAXIT,DIRALL,FILEO
2010   format(/'--- PARAMETER MODIFICATION MENU ---',//
     2    'Enter character of item to modify:',//
     3    '  <S> for START node, currently at (',i3,',',i3,',',i3,')',/
     4    '  <G> for GOAL  node, currently at (',i3,',',i3,',',i3,')',/
     5    '  <X> for OBSX node  (isolated) , currently =',f8.4,/
     6    '  <C> for OBSC node  (connected), currently =',f8.4,/
     7    '  <B> for B=GND (start node) pot, currently =',f8.4,/
     8    '  <V> for VCC   (goal node)  pot, currently =',f8.4,/
     9    '  <N> for max NODAL change allowed per iter =',f8.4,/
     +    '  <I> for max number of ITERATIONS allowed  =',i8,/
     1    '  <D> for horiz move-DIRECTIONS  (4 or 8)   =',i8,/
     2    '  <M> for MAZE element positioning',/
     3    '  <R> for RE-SHOW maze environment',/
     4    '  <O> for OUTPUT file_name,  currently :  ',a12,//
     5    '  <Q> or <CR> to  QUIT modification menu !')
      call bell(1)
      write(*,2020)
```

```
2020   format('CHOICE : '$)
       read(*,'(a1)') MOD
       if (MOD.eq.'Q' .or. MOD.eq.'q' .or. MOD.eq.' ') then
         goto 8000
       else if (MOD.eq.'S' .or. MOD.eq.'s') then
         call bell(1)
         write(*,4010)
4010     format ('Enter new Start ROW,COL,DEP :  '$)
         read(*,*) NROW,NCOL,NDEP
         if (NROW.lt.1 .or. NROW.gt.ROW .or.
     2        NCOL.lt.1 .or. NCOL.gt.COL .or.
     3        NDEP.lt.1 .or. NDEP.gt.DEP) then
           write(*,*)
           write(*,*)'Node mu    between  (  1 - ',ROW,
     +             ',  1 - ',COL,',  1 - ',DEP,')'
           write(*,*)
           write(*,*)
           goto 1000
         end if
         STARTN(1)=NROW
         STARTN(2)=NCOL
         STARTN(3)=NDEP
         write(*,*)
         write(*,*)
       else if (MOD.eq.'G' .or. MOD.eq.'g') then
         call bell(1)
         write(*,4020)
4020     format ('Enter new Goal ROW,COL,DEP :  '$)
         read(*,*) NROW,NCOL,NDEP
         if (NROW.lt.1 .or. NROW.gt.ROW .or.
     2        NCOL.lt.1 .or. NCOL.gt.COL .or.
     3        NDEP.lt.1 .or. NDEP.gt.DEP) then
           write(*,*)
           write(*,*)'Node must be between  (  1 - ',ROW,
     +             ',  1 - ',COL,',  1 - ',DEP,')'
           write(*,*)
           write(*,*)
           goto 1000
         end if
         GOALN(1)=NROW
         GOALN(2)=NCOL
         GOALN(3)=NDEP
         write(*,*)
         write(*,*)
       else if (MOD.eq.'X' .or. MOD.eq.'x') then
         call bell(1)
         write(*,4030)
4030     format ('Enter new OBSX value :  '$)
         read(*,*) OBSX
         write(*,*)
         write(*,*)
       else if (MOD.eq.'C' .or. MOD.eq.'c') then
         call bell(1)
         write(*,4040)
4040     format ('Enter new OBSC value :  '$)
         read(*,*) OBSC
         write(*,*)
         write(*,*)
       else if (MOD.eq.'B' .or. MOD.eq.'b') then
         call bell(1)
         write(*,4050)
4050     format ('Enter new GND value :  '$)
```

```
          read(*,*) GND
          write(*,*)
          write(*,*)
        else if (MOD.eq.'V' .or. MOD.eq.'v') then
          call bell(1)
          write(*,4060)
4060      format ('Enter new VCC value :  '$)
          read(*,*) VCC
          write(*,*)
          write(*,*)
        else if (MOD.eq.'N' .or. MOD.eq.'n') then
          call bell(1)
          write(*,4070)
4070      format ('Enter new MAX NODAL value :  '$)
          read(*,*) ALLERR
          write(*,*)
          write(*,*)
        else if (MOD.eq.'I' .or. MOD.eq.'i') then
          call bell(1)
          write(*,4080)
4080      format ('Enter new MAX # ITER value :  '$)
          read(*,*) MAXIT
          write(*,*)
          write(*,*)
        else if (MOD.eq.'M' .or. MOD.eq.'m') then
          call bell(1)
          write(*,4090)
4090      format ('Enter ROW,COL,DEP of node to be changed :  '$)
          read(*,*) NROW,NCOL,NDEP
          if (NROW.lt.1 .or. NROW.gt.ROW .or.
     2         NCOL.lt.1 .or. NCOL.gt.COL .or.
     3         NDEP.lt.1 .or. NDEP.gt.DEP) then
            write(*,*)
            write(*,*)'Node must be between  (  1 - ',ROW,
     +             ',  1 - ',COL,',  1 - ',DEP,')'
            write(*,*)
            write(*,*)
            goto 1000
          end if
          write(*,*)
          write(*,4100)
4100      format ('Enter CHARacteristic for new node :  '$)
          read(*,'(a1)') NCHAR
          write(*,*)
          write(*,*)
          if(NCHAR.ne.OBSTX .and. NCHAR.ne.OBSTC .and.
     2       NCHAR.ne.FREE1 .and. NCHAR.ne.FREE2 .and.
     3       NCHAR.ne.FREE3) then
            write(*,*)'CHARacter must be ',OBSTX,' , ',OBSTC,
     +                ' , ',FREE1,' , ',FREE2,' , or ',FREE3
            write(*,*)
            write(*,*)
          else
            MAZ(NROW,NCOL,NDEP)=NCHAR
          end if
        else if (MOD.eq.'R' .or. MOD.eq.'r') then
          write(*,*)
          call MAZOUT(SCR)
          if (SCR.eq.9) call bell(1)
          if (SCR.eq.9) pause 'press <CR> to continue... '
          if (SCR.eq.9) write(*,*)
        else if (MOD.eq.'D' .or. MOD.eq.'d') then
```

```
          call bell(1)
          write(*,4110)
4110      format ('Enter new horiz move-DIRECTIONS allowed :  '$)
          read(*,*) DIRALL
          write(*,*)
          write(*,*)
          if (DIRALL.ne.8 .and. DIRALL.ne.4) then
            write(*,*)'DIRALL must be 4 or 8'
            DIRALL=4
            write(*,*)
            write(*,*)
          end if
        else if (MOD.eq.'O' .or. MOD.eq.'o' .or. MOD.eq.'0') then
          call bell(1)
          write(*,4120)
4120      format ('Enter new OUTPUT file_name :  '$)
          read(*,*) FILEO
          write(*,*)
          write(*,*)
        else
          write(*,*) MOD,' is not a valid OPTION...'
          write(*,*)
          write(*,*)
        end if
7000      goto 1000
8000    if (STARTN(1).lt.1 .or. STARTN(1).gt.ROW .or.
     2       STARTN(2).lt.1 .or. STARTN(2).gt.COL .or.
     3       STARTN(3).lt.1 .or. STARTN(3).gt.DEP) then
          write(*,*)
          write(*,*)'Start node must be between  (  1 - ',ROW,
     +             ',  1 - ',COL,',  1 - ',DEP,')'
          write(*,*)
          write(*,*)
          goto 1000
        end if
        if (GOALN(1).lt.1 .or. GOALN(1).gt.ROW .or.
     2       GOALN(2).lt.1 .or. GOALN(2).gt.COL .or.
     3       GOALN(3).lt.1 .or. GOALN(3).gt.DEP) then
          write(*,*)
          write(*,*)'Goal node must be between  (  1 - ',ROW,
     +             ',  1 - ',COL,',  1 - ',DEP,')'
          write(*,*)
          write(*,*)
          goto 1000
        end if
c  set weights array
        do 30 K=0,DEP+1
          do 20 J=0,COL+1
            do 10 I=0,ROW+1
              if (MAZ(I,J,K).eq.FREE1 .or. MAZ(I,J,K).eq.FREE2 .or.
     +             MAZ(I,J,K).eq.FREE3 .or. MAZ(I,J,K).eq.OBSTC) then
                WTS(I,J,K)=1
              else
                WTS(I,J,K)=0
              end if
10          continue
20        continue
30      continue
c  open output file
        if (SCR.eq.9) then
          open(FIL,file=FILEO, status='NEW')
        else
```

```
         open(FIL,file=FILEO)
       endif
       write(FIL,2030) FILEO
2030   form<  ('file: ',a12//)
       call MAZLGO(FIL)
c  add start/goal data to MAZ
       MAZ(STARTN(1),STARTN(2),STARTN(3))=START
       MAZ(GOALN(1),GOALN(2),GOALN(3))=GOAL

c  add start/goal data to WTS

       WTS(STARTN(1),STARTN(2),STARTN(3))=1
       WTS(GOALN(1),GOALN(2),GOALN(3))=1
8888   return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine MAZPOT
c  converts info from char array MAZ to value array POT, + init PFLD
c declaration of global/common variables
       integer       SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
       integer       MAXIT,CNT,MOVE,I,J,K,DIRALL
       real          OBSX,OBSC,GND,VCC,DIST
       real          ALLERR,ACTERR,TTLERR
       integer       WTS(0:81,0:81,0:9)
       real          POT(0:81,0:81,0:9)
       real          PFLD(0:81,0:81,0:9)
       integer       PATH(0:1000,4)
       character*12  FILEDF,FILEP,FILEM,FILEO
       character*1   MAZ(0:81,0:81,0:9)
       character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
       common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2                CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                ALLERR,ACTERR,TTLERR,WTS,
     4                POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
       character*1   TEMP
        integer      U
        do 40 K=0,DEP+1
          do 30 I=0,ROW+1
            do 20 J=0,COL+1
              TEMP=MAZ(I,J,K)
              if      (TEMP.eq.OBSTX) then
                POT(I,J,K) = OBSX
              else if (TEMP.eq.OBSTC) then
                POT(I,J,K) = OBSC
              else if (TEMP.eq.FREE1 .or. TEMP.eq.FREE2 .or.
     +                 TEMP.eq.FREE3)  then
                POT(I,J,K) = GND
              else if (TEMP.eq.START) then
                POT(I,J,K) = GND
              else if (TEMP eq.GOAL)  then
                 POT(I,J,K) = VCC
              else
                POT(I,J,K)=OBSX
                do 10 U=FIL,SCR ,SCR-FIL
                  write(U,*)
                  write(U,2000) TEMP,I,J,K
2000              format(/'Error while assigning values to maze,',/
     2                    'unknown character entry : ',a1,
     3                    ' , at (',i3,',',i3,',',i3,')',/
     4                   '(entry set to isolated obstacle value)'//)
10               continue
               end if
20             continue
30            continue
40          continue
c  initialize array PFLD outer boundaries to obstacle values
c  init top and bottom of array
        do 60 J=0,COL+1
          do 50 I=0,ROW+1
            PFLD(I,J,0)=OBSX
            PFLD(I,J,DEP+1)=OBSX
50        continue
60      continue
c  init sides of array
        do 90 K=1,DEP
          do 70 J=0,COL+1
            PFLD(0,J,K)=OBSX
            PFLD(ROW+1,J,K)=OBSX
70        continue
          do 80 I=1,ROW
            PFLD(I,0,K)=OBSX
            PFLD(I,COL+1,K)=OBSX
80        continue
90      continue
         return
         end

c----+---1---------2---------3---------4---------5---------6---------7--

        subroutine MAZFLD
c  calculate final values for potential field array
c  and show in progress error calculations
c declaration of global/common variables
        integer       SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
        integer       MAXIT,CNT,MOVE,I,J,K,DIRALL
        real          OBSX,OBSC,GND,VCC,DIST
        real          ALLERR,ACTERR,TTLERR
        integer       WTS(0:81,0:81,0:9)
        real          POT(0:81,0:81,0:9)
        real          PFLD(0:81,0:81,0:9)
        integer       PATH(0:1000,4)
        character*12  FILEDF,FILEP,FILEM,FILEO
        character*1   MAZ(0:81,0:81,0:9)
        character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
        common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2                 CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                 ALLERR,ACTERR,TTLERR,WTS,
     4                 POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                 MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
        integer       PINTVL,PRNCNT,ITGLOP
        real          SUM,ERROR,INVSR2,SNWTS
        character*1   GBLANS
        INVSR2 =1.0 / sqrt(2.0)
        ITGLOP=0
        PINTVL=10
        write(*,*)
        write(*,*) 'Node Potential Array Iteration/Error Status: '
        write(*,*)
        CNT=1
        PRNCNT=PINTVL-1
10       ACTERR=0.0
        TTLERR=0.0
        do 60 K=1,DEP
          do 50 I=1,ROW
            do 40 J=1,COL
              if (MAZ(I,J,K).eq.FREE1 .or. MAZ(I,J,K).eq.FREE2 .or.
     +             MAZ(I,J,K).eq.FREE3) then
                if(DIRALL.eq.4) then
                  SNWTS = WTS(I-1,J,K) + WTS(I,J+1,K) + WTS(I+1,J,K)
     +                 + WTS(I,J-1,K) + WTS(I,J,K-1) + WTS(I,J,K+1)
                 else
                  SNWTS = WTS(I-1,J,K) + WTS(I,J+1,K) + WTS(I+1,J,K)
     2                 + WTS(I,J-1,K) + WTS(I,J,K-1) + WTS(I,J,K+1)
     3                 + ( WTS(I-1,J+1,K) + WTS(I+1,J+1,K)
     4                   + WTS(I+1,J-1,K) + WTS(I-1,J-1,K)) * INVSR2
               endif
                if (SNWTS .eq. 0.0) then
                  PFLD(I,J,K) = POT(I,J,K)
                else
                  if(DIRALL.eq.4) then
                   SUM = POT(I-1,J,K) * WTS(I-1,J,K)
     2                 + POT(I,J+1,K) * WTS(I,J+1,K)
     3                 + POT(I+1,J,K) * WTS(I+1,J,K)
     4                 + POT(I,J-1,K) * WTS(I,J-1,K)
     5                 + POT(I,J,K-1) * WTS(I,J,K-1)
     6                 + POT(I,J,K+1) * WTS(I,J,K+1)
                  else
                   SUM = POT(I-1,J,K) * WTS(I-1,J,K)
     2                 + POT(I,J+1,K) * WTS(I,J+1,K)
     3                 + POT(I+1,J,K) * WTS(I+1,J,K)
     4                 + POT(I,J-1,K) * WTS(I,J-1,K)
     5                 + POT(I,J,K-1) * WTS(I,J,K-1)
     6                 + POT(I,J,K+1) * WTS(I,J,K+1)
     7                 + ( POT(I-1,J+1,K) * WTS(I-1,J+1,K)
     8                   + POT(I+1,J+1,K) * WTS(I+1,J+1,K)
     9                   + POT(I+1,J-1,K) * WTS(I+1,J-1,K)
     +                   + POT(I-1,J-1,K) * WTS(I-1,J-1,K)) * INVSR2
                 endif
                 PFLD(I,J,K) = SUM / SNWTS
                end if
                ERROR=ABS(PFLD(I,J,K)-POT(I,J,K))
                TTLERR=TTLERR+ERROR
                if (ERROR.gt.ACTERR) ACTERR=ERROR
c  check if neighbors of START PT show "potential change"
c  thus indicating first path found between GOAL and START!!!
              else if (ITGLOP.eq.0 .and. MAZ(I,J,K).eq.START) then
                PFLD(I,J,K)=POT(I,J,K)
                if(DIRALL.eq.4) then
                 SUM = POT(I-1,J,K) * WTS(I-1,J,K)
     2               + POT(I,J+1,K) * WTS(I,J+1,K)
     3               + POT(I+1,J,K) * WTS(I+1,J,K)
     4               + POT(I,J-1,K) * WTS(I,J-1,K)
     5               + POT(I,J,K-1) * WTS(I,J,K-1)
     6               + POT(I,J,K+1) * WTS(I,J,K+1)
                else
                 SUM = POT(I-1,J,K) * WTS(I-1,J,K)
     2               + POT(I,J+1,K) * WTS(I,J+1,K)
     3               + POT(I+1,J,K) * WTS(I+1,J,K)
     4               + POT(I,J-1,K) * WTS(I,J-1,K)
     5               + POT(I,J,K-1) * WTS(I,J,K-1)
     6               + POT(I,J,K+1) * WTS(I,J,K+1)
     7            + ( POT(I-1,J+1,K) * WTS(I-1,J+1,K)
     8              + POT(I+1,J+1,K) * WTS(I+1,J+1,K)
     9              + POT(I+1,J-1,K) * WTS(I+1,J-1,K)
     +              + POT(I-1,J-1,K) * WTS(I-1,J-1,K)) * INVSR2
              endif
               if (SUM.gt.GND) then
                  ITGLOP=CNT
                 write(SCR,2000) ITGLOP
                 write(FIL,2000) ITGLOP
2000             format(//'Global minimal distance solution found ',
     +                  'using',i5,' iterations!!!'/)
                  call bell(1)
                  write(*,2010)
2010              format('Enter <Y> to continue iterating, or',/6x,
     +              '<CR> to compute path with current nodal values:'$)
                  GBLANS=' '
                  read(*,'(a1)') GBLANS
                  write(*,*)
                  if (GBLANS.ne.'Y' .and. GBLANS.ne.'y') goto 100
               endif
             else
               PFLD(I,J,K)=POT(I,J,K)
             end if
40         continue
50       continue
60     continue
       do 90 K=0,DEP+1
         do 80 I=0,ROW+1
           do 70 J=0,COL+1
             POT(I,J,K)=PFLD(I,J,K)
70          continue
80       continue
90     continue
       PRNCNT=PRNCNT+1
       if (PRNCNT.eq.PINTVL) then
         write(*,2020)CNT,ACTERR,TTLERR
2020     format('Iter:',i4,5x,'Max nodal chg = ',f7.4,
     +           5x,'Total iter chg = ',f8.4)
         PRNCNT=0
         if (CNT.eq.1) PRNCNT=1
       end if
       if (CNT.ge.MAXIT) goto 100
       CNT=CNT+1
       if (ACTERR.gt.ALLERR) goto 10
100    return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine MAZFNL(U)
c      outputs potential field
c declaration of global/common variables
       integer        SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
       integer        MAXIT,CNT,MOVE,I,J,K,DIRALL
       real           OBSX,OBSC,GND,VCC,DIST
       real           ALLERR,ACTERR,TTLERR
       integer        WTS(0:81,0:81,0:9)
       real           POT(0:81,0:81,0:9)
       real           PFLD(0:81,0:81,0:9)
       integer        PATH(0:1000,4)
       character*12   FILEDF,FILEP,FILEM,FILEO
       character*1    MAZ(0:81,0:81,0:9)
```

```
      character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
      common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2               CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3               ALLERR,ACTERR,TTLERR,WTS,
     4               POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5               MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
      integer       U
c  output potentials array
      write(U,*)
      write(U,*) 'Final Nodal Potentials... '
      write(U,*)
      do 20, K=1,DEP
        if (DEP.ne.1) then
          write(U,*)
          write(U,*) 'Depth Layer:',K
        endif
        do 10, I=1,ROW
          write(U,2000) (POT(I,J,K),J=1,COL)
2000      format (128e11.3)
10      continue
20    continue
      write (U,*)
      return
      end

c----+---1---------2---------3---------4---------5---------6---------7--

      subroutine MAZMOV
c     calculate path
c declaration of global/common variables
      integer       SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
      integer       MAXIT,CNT,MOVE,I,J,K,DIRALL
      real          OBSX,OBSC,GND,VCC,DIST
      real          ALLERR,ACTERR,TTLERR
      integer       WTS(0:81,0:81,0:9)
      real          POT(0:81,0:81,0:9)
      real          PFLD(0:81,0:81,0:9)
      integer       PATH(0:1000,4)
      character*12  FILEDF,FILEP,FILEM,FILEO
      character*1   MAZ(0:81,0:81,0:9)
      character*1   OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
      common        SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2               CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3               ALLERR,ACTERR,TTLERR,WTS,
     4               POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5               MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
      integer       L,RC,CLOC(3),NLOC(3),MKR
      real          SR2,CNODE,DN,DE,DS,DW,DU,DD,DNE,DSE,DSW,DNW,MAXD
      SR2=sqrt(2.0)
c  initialize array PATH to all zeros
      do 20 RC=1,4
        do 10 L=1,1000
          PATH(L,RC)=0
10      continue
20    continue
c set MOVE to 0, and MKR to ASCII char code for '0'
      MOVE=0
      DIST=0.0
      MKR=48
c  set current location
```

```
        CLOC(1)=STARTN(1)
        CLOC(2)=STARTN(2)
        CLOC(3)=STARTN(3)
c  set path start
        PATH(0,1)=STARTN(1)
        PATH(0,2)=STARTN(2)
        PATH(0,3)=STARTN(3)
30      if (CLOC(1).eq.GOALN(1) .and. CLOC(2).eq.GOALN(2)
     +       .and. CLOC(3).eq.GOALN(3)) then
          PATH(MOVE,4)=0
          goto 8888
        end if
        CNODE = PFLD(CLOC(1),CLOC(2),CLOC(3))
c  cal delta potentials for North, East, South, West, Up, & Down
        DN  = PFLD(CLOC(1)-1,CLOC(2),CLOC(3))-CNODE
        DE  = PFLD(CLOC(1),CLOC(2)+1,CLOC(3))-CNODE
        DS  = PFLD(CLOC(1)+1,CLOC(2),CLOC(3))-CNODE
        DW  = PFLD(CLOC(1),CLOC(2)-1,CLOC(3))-CNODE
        DU  = PFLD(CLOC(1),CLOC(2),CLOC(3)-1)-CNODE
        DD  = PFLD(CLOC(1),CLOC(2),CLOC(3)+1)-CNODE
        if (DIRALL.eq.4) goto 40
c  calculate delta potentials for NE, SE, SW, & NW
        DNE = (PFLD(CLOC(1)-1,CLOC(2)+1,CLOC(3))-CNODE) / SR2
        DSE = (PFLD(CLOC(1)+1,CLOC(2)+1,CLOC(3))-CNODE) / SR2
        DSW = (PFLD(CLOC(1)+1,CLOC(2)-1,CLOC(3))-CNODE) / SR2
        DNW = (PFLD(CLOC(1)-1,CLOC(2)-1,CLOC(3))-CNODE) / SR2
        MAXD = MAX(DN,DE,DS,DW,DU,DD,DNE,DSE,DSW,DNW)
        goto 50
40      MAXD = MAX(DN,DE,DS,DW,DU,DD)
c  check for no solution
50      if (MAXD .le. 0.0) then
          write(SCR,2000)
          write(FIL,2000)
2000      format(/'!!     No solution found for this problem     !!',/
     +            '(allow more iterations or look for blocked path)'//)
          PATH(MOVE,4)= -1
          goto 8888
        end if
c  check N move
        if (MAXD.eq.DN) then
          NLOC(1)=CLOC(1)-1
          NLOC(2)=CLOC(2)
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=1
          DIST=DIST+1.0
          goto 60
c  check E move
        else if (MAXD.eq.DE) then
          NLOC(1)=CLOC(1)
          NLOC(2)=CLOC(2)+1
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=2
          DIST=DIST+1.0
          goto 60
c  check S move
        else if (MAXD.eq.DS) then
          NLOC(1)=CLOC(1)+1
          NLOC(2)=CLOC(2)
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=3
          DIST=DIST+1.0
          goto 60
```

```
c   check W move
        else if (MAXD.eq.DW) then
          NLOC(1)=CLOC(1)
          NLOC(2)=CLOC(2)-1
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=4
          DIST=DIST+1.0
          goto 60
c   check U move
        else if (MAXD.eq.DU) then
          NLOC(1)=CLOC(1)
          NLOC(2)=CLOC(2)
          NLOC(3)=CLOC(3)-1
          PATH(MOVE,4)=10
          DIST=DIST+1.0
          goto 60
c   check D move
        else if (MAXD.eq.DD) then
          NLOC(1)=CLOC(1)
          NLOC(2)=CLOC(2)
          NLOC(3)=CLOC(3)+1
          PATH(MOVE,4)=20
          DIST=DIST+1.0
          goto 60
        end if
c   check NE move
        if (MAXD.eq.DNE) then
          NLOC(1)=CLOC(1)-1
          NLOC(2)=CLOC(2)+1
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=5
          DIST=DIST+SR2
c   check SE move
        else if (MAXD.eq.DSE) then
          NLOC(1)=CLOC(1)+1
          NLOC(2)=CLOC(2)+1
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=6
          DIST=DIST+SR2
c   check SW move
        else if (MAXD.eq.DSW) then
          NLOC(1)=CLOC(1)+1
          NLOC(2)=CLOC(2)-1
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=7
          DIST=DIST+SR2
c   check NW move
        else if (MAXD.eq.DNW) then
          NLOC(1)=CLOC(1)-1
          NLOC(2)=CLOC(2)-1
          NLOC(3)=CLOC(3)
          PATH(MOVE,4)=8
          DIST=DIST+SR2
        end if
60      MOVE=MOVE+1
        MKR=MKR+1
        if (MKR.eq.59) MKR=48
        CLOC(1)=NLOC(1)
        CLOC(2)=NLOC(2)
        CLOC(3)=NLOC(3)
        PATH(MOVE,1)=CLOC(1)
        PATH(MOVE,2)=CLOC(2)
```

```
       PATH(MOVE,3)=CLOC(3)
       if (CLOC(1).ne.GOALN(1) .or. CLOC(2).ne.GOALN(2)
     2     .or. CLOC(3).ne.GOALN(3))
     3    MAZ(CLOC(1),CLOC(2),CLOC(3)) = char(MKR)
       goto 30
8888   return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       subroutine MAZPTH(U)
c  output path
c declaration of global/common variables
       integer        SCR,FIL,ROW,COL,DEP,STARTN(3),GOALN(3)
       integer        MAXIT,CNT,MOVE,I,J,K,DIRALL
       real           OBSX,OBSC,GND,VCC,DIST
       real           ALLERR,ACTERR,TTLERR
       integer        WTS(0:81,0:81,0:9)
       real           POT(0:81,0:81,0:9)
       real           PFLD(0:81,0:81,0:9)
       integer        PATH(0:1000,4)
       character*12   FILEDF,FILEP,FILEM,FILEO
       character*1    MAZ(0:81,0:81,0:9)
       character*1    OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
       common         SCR,FIL,ROW,COL,DEP,STARTN,GOALN,MAXIT,
     2                 CNT,MOVE,I,J,K,DIRALL,OBSX,OBSC,GND,VCC,DIST,
     3                 ALLERR,ACTERR,TTLERR,WTS,
     4                 POT,PFLD,FILEDF,FILEP,FILEM,FILEO,PATH,
     5                 MAZ,OBSTX,OBSTC,FREE1,FREE2,FREE3,START,GOAL
c  declare local variables
       integer        STEP,U
       character*4    MAZDIR
       write(U,2000) MOVE,DIST
2000   format(/,'Solution Path... ',
     2           '    Path =',i4,' steps,',f8.2,' units',//
     3           '   Step         Current Node        Next Move',/
     4           '  Number     ( Row, Col, Depth)     Direction'/)
c  output STEP   ROW   COL  DEPTH  DIRECTION
       write(U,2010) ' START PT',PATH(0,1),PATH(0,2),PATH(0,3),
     +                 MAZDIR(PATH(0,4))
2010   format   (a9,3x,'(',i3,' ,',i3,' ,',i3,'   )',6x,a4)
       if (MOVE.eq.0) goto 20
       do 10 STEP=1,MOVE
         write(U,2020) STEP,PATH(STEP,1),PATH(STEP,2),PATH(STEP,3),
     +                   MAZDIR(PATH(STEP,4))
2020     format   (i5,7x,'(',i3,' ,',i3,' ,',i3,'   )',6x,a4)
10     continue
20     write(U,*)
       return
       end

c----+---1---------2---------3---------4---------5---------6---------7--

       function MAZDIR(IDIR)
c  produces direction character string from integer input
       character*4   MAZDIR
       integer       IDIR
       if      (IDIR.eq.1) then
         MAZDIR='  N '
       else if (IDIR.eq.2) then
         MAZDIR='  E '
       else if (IDIR.eq.3) then
```

```
          MAZDIR='  S '
        else if (IDIR.eq.4) then
          MAZDIR='  W '
        else if (IDIR.eq.5) then
          MAZDIR=' NE '
        else if (IDIR.eq.6) then
          MAZDIR=' SE '
        else if (IDIR.eq.7) then
          MAZDIR=' SW '
        else if (IDIR.eq.8) then
          MAZDIR=' NW '
        else if (IDIR.eq.10) then
          MAZDIR=' UP '
        else if (IDIR.eq.20) then
          MAZDIR='DOWN'
        else if (IDIR.eq.0) then
          MAZDIR='GOAL'
        else
          MAZDIR=' ?? '
        end if
        return
        end

c----+---1---------2---------3---------4---------5---------6---------7--

        subroutine bell(NUM)
c  rings bell NUM times
c  declare local variables
        integer       NUM,RINGS,DELAY
c  ASCII code 7 = bell ring
        do 20 RINGS=1,NUM
          write(*,*) char(7)
c  delay loop
          do 10 DELAY=1,8000
10        continue
20      continue
        return
        end
```

amaz3d.prf
(preferences data file)

```
9                    SCR     (screen device output number) (use 6 for SUNs)
bldgnav.par          FILEDF  (default input parameter file-name)
```

Appendix G

AMAZ3D Test Data and Output

The fcllowing output listings show some of the capabilities of the AMAZ3D program. The first two example problems, comprised of six files total, arc complete: 1) **maz.par** and 2) **maz.dat** are the parameter and sensory data files for the same two-dimensional (2D) maze solved by the Maze Machine in Appendix E as Test 4, and 3) **maz.out** is the AMAZ3D output 'isting for this problem (Note the similarity in final nodal values for the connectionist network and the identical path solutions.), 4) **maz3d.par,** and 5) **maz3d.dat** are the parameter and sensory data files for a three-dimensional (3D) 7 row, by 7 column, by 7 layer maze, and 6) **maz3d.out** is the program's output listing for this example.

In an effort at brevity, no other parameter or sensory data files, nor the AMAZ3D standard logo at the beginning of output listings, are included. Since all the pertinent parameter and sensory data information is included in the program's output listing, no loss of understanding should occur. The remaining sample output listings and their significance are as follows:

File: **mazno.out** shows a 7 by 7, 2D maze which results in a non-optimal (in terms of minimal distance) solution path. The reason for the path diversion is a fork in the shortest distance path which combines with an alternative path near the start to cause the alternate (longer) path to initially show a greater nodal value increase for the first step East as opposed to West. (Remember that the solution path is based on locally optimal moves.)

File: **maznoc.out** (using iteration cut-off feature) shows the same 7 by 7, 2D maze as mazno.out, however this time the feature of iteration cut-off for faster solutions is implemented. (Note a path length of 14 units as opposed to 16 for the mazno.out solution. This solution happens to provide the optimal path for this problem.)

File: **mazn.out** shows a 15 by 17, 2D maze which results in an optimal (in terms of minimal distance) solution path.

File: **maznc.out** (using iteration cut-off feature) shows the same 15 by 17, 2D maze as mazn.out, however this time the feature of iteration cut-off for faster solutions is implemented. (Note a path length of 45 units as opposed to 43 for the mazn.out solution.) This solution happens to be sub-optimal. The reason for the path diversion is the large free space near / on the optimal path which causes the nodes in this region to have smaller nodal values during early iterations due to dispersion, while the isolated longer path is not influenced by any neighboring free space nodes.

File: **landnav.out** shows a 44 by 64, 2D land navigation problem implementing the connected obstacles feature. The data for this example was provided by Peter Weiland from research he conducted on using scanning lasers for robotic navigation [see Weiland (1989)]. The data depicts processed local terrain sensory input from a cross-country mobile robot. In this example, it is assumed to be desirable to avoid the two 'round' obstacles and the West side wall while traveling from start to goal. Here, connected obstacles are obstacles to be avoided, while isolated obstacles represent shadow regions. (Note the different behavior due to the connected obstacles.) The extension of using connected obstacles as 'repulsive potential fields' was inspired by the work done by Norwood (1989).

File: **landnavc.out** (using iteration cut-off) shows the same 44 by 64, 2D maze as landnav.out, however this time the feature of iteration cut-off is implemented. (Notes: 1) Diagonal moves have a unit value of -> square root of 2. 2) This run results in a path length of 44.14 units as opposed to 46.63 for the landnav.out solution.)

File: **bldgnav.out** shows a 30 by 80, 2D building interior navigation problem. The objective is to efficiently move from one room to another while travelling through hallways as necessary. Doors are shown differently from regular free space nodes to highlight the fact that this program could be easily used to control a mobile robot where the program calls on additional routines which allow the robot to properly open and close doors during its travels.

File: **b3dnav.out** shows a 3D problem similar to the 30 by 80 building interior, of bldgnav.out, however this time a second story is implemented through the addition of an elevator in the North-East corner of the floorplan. Layer 2 is used as a divider between the first and second stories/ floors, thus causing this implementation to use 3 layers.

maz.par

```
7              ROW            (maze i dim)
7              COL            (maze j dim)
1              DEP            (maze k dim)
X              OBSTX          (character representation)
@              OBSTC          (character representation)
.              FREE1          (character representation)
D              FREE2          (character representation)
E              FREE3          (character representation)
S              START          (character representation)
G              GOAL           (character representation)
7              STARTN(i)      (start i coord)
2              STARTN(j)      (  "   j coord)
1              STARTN(k)      (  "   k coord)
1              GOALN(i)       ( goal i coord)
7              GOALN(j)       (  "   j coord)
1              GOALN(k)       (  "   k coord)
0.0            OBSX           (forced potential value for ObstX)
0.0            OBSC           (forced potential value for ObstC)
0.0            GND            (forced potential value for Start)
10.0           VCC            (forced potential value for Goal )
0.0001         ALLERR         (max node change/iter allowed)
3000           MAXIT          (max  # of iterations allowed)
maz.dat        FILEM          (maze input file)  (12 char max)
maz.out        FILEO          (data output file) (12 char max)
4              DIRALL         (2D move-directions allowed, 4 or 8)
```

Maz.dat

```
X......
..X.XXX
.X....X
XXX.X..
....XX.
.XX.X..
.......
```

file: maz.out

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
%      A      M     M       A      ZZZZZ   333   DDDD    %
%     A A     MM   MM      A A       Z       3   D   D   %
%    AAAAA    M M M M     AAAAA     Z       33   D   D   %
%   A     A   M  M  M    A     A   Z         3   D   D   %
%  A       A  M     M   A       A  ZZZZZ   333   DDDD    %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Chris Schuster     MEMS/Robotics,Rice University     v6.03


Maze Environment...

  The Maze:   maz.dat      has   7 Rows ,  7 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated)     @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  7,  2,  1)
        Goal  node = G , at (  1,  7,  1)

      1234567

  1   X.....G
  2   ..X.XXX
  3   .X....X
  4   XXX.X..
  5   ....XX.
  6   .XX.X..
  7   .S.....

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =   0.0000
  Connected obstacle value  (ObsC)  =   0.0000
  Start node value  - - - -  (Gnd)  =   0.0000
  Goal  node value  - - - -  (Vcc)  =  10.5300
  Max nodal change allowed per iter =   0.0010
  Max number of iterations allowed  =     2000
  Horiz directions allowed (4 or 8) =        4


Global minimal distance solution found using   11 iterations!!!


Final Nodal Potentials...

  0.000E+00  0.682E+01  0.682E+01  0.683E+01  0.806E+01  0.930E+01  0.105E+02
  0.682E+01  0.682E+01  0.000E+00  0.560E+01  0.000E+00  0.000E+00  0.000E+00
  0.682E+01  0.000E+00  0.437E+01  0.437E+01  0.405E+01  0.374E+01  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.346E+01  0.000E+00  0.343E+01  0.311E+01
  0.127E+01  0.170E+01  0.212E+01  0.255E+01  0.000E+00  0.000E+00  0.280E+01
  0.847E+00  0.000E+00  0.000E+00  0.206E+01  0.000E+00  0.234E+01  0.249E+01
  0.424E+00  0.000E+00  0.788E+00  0.158E+01  0.188E+01  0.219E+01  0.234E+01
```

```
                Total Iterations  =     371
Maximum individual nodal change   =  0.0010
   Total iteration nodal change   =  0.0127


Solution Path...     Path =  11 steps,   11.00 units

   Step        Current Node       Next Move
  Number    ( Row, Col, Depth)    Direction

 START PT   (  7 ,  2 ,  1   )        E
    1       (  7 ,  3 ,  1   )        E
    2       (  7 ,  4 ,  1   )        N
    3       (  6 ,  4 ,  1   )        N
    4       (  5 ,  4 ,  1   )        N
    5       (  4 ,  4 ,  1   )        N
    6       (  3 ,  4 ,  1   )        N
    7       (  2 ,  4 ,  1   )        N
    8       (  1 ,  4 ,  1   )        E
    9       (  1 ,  5 ,  1   )        E
   10       (  1 ,  6 ,  1   )        E
   11       (  1 ,  7 ,  1   )       GOAL

Maze Solution...

  The Maze:   maz.dat      has   7 Rows ,   7 Cols ,   1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,    @ (connected)
        Free Space = .    [ + D (doors) ,    E (elevators) ]
        Start node = S , at (  7,   2,   1)
        Goal  node = G , at (  1,   7,   1)

      1234567

  1   X..890G
  2   ..X7XXX
  3   .X.6..X
  4   XXX5X..
  5   ...4XX.
  6   .XX3X..
  7   .S12...
```

maz.3d.par

```
7              ROW           (maze i dim)
7              COL           (maze j dim)
7              DEP           (maze k dim)
X              OBSTX         (character representation)
@              OBSTC         (character representation)
.              FREE1         (character representation)
D              FREE2         (character representation)
E              FREE3         (character representation)
S              START         (character representation)
G              GOAL          (character representation)
3              STARTN(i)     (start i coord)
3              STARTN(j)     (  "   j coord)
1              STARTN(k)     (  "   k coord)
3              GOALN(i)      ( goal i coord)
5              GOALN(j)      (  "   j coord)
7              GOALN(k)      (  "   k coord)
0.0            OBSX          (forced potential value for ObstX)
0.0            OBSC          (forced potential value for ObstC)
0.0            GND           (forced potential value for Start)
10.0           VCC           (forced potential value for Goal )
0.01           ALLERR        (max node change/iter allowed)
150            MAXIT         (max  # of iterations allowed)
maz3d.dat      FILEM         (maze input file)  (12 char max)
maz3d.out      FILEO         (data output file) (12 char max)
4              DIRALL        (2D move-directions allowed, 4 or 8)
```

maz3d.dat

```
1
...X...
XX.XXXX
...X...
XXXX.XX
.X.X...
.X.XXXX
.X.....
2
.XXX.X.
XXXXXXX
.XXXXX.
XXXXXXX
.X.XXX.
XXXXXXX
XXXXXX.
3
.X.X.X.
XX.X.X.
.X.X.X.
.X.XXXX
.X.X...
XXXXXXX
.......
4
.X.XXXX
XXXXXXX
XXXX.XX
XXXXXXX
XXXX.XX
XXXXXXX
.XXXXXX
5
.X.X.X.
.X.X.X.
...X.X.
XXXXXX.
...X.X.
.XXXXXX
.X.....
6
XXXX.X.
XXXXXXX
XXXXXXX
XXXYXXX
XX.X.X.
XXXXXXX
XX.XXX.
7
.....X.
.XXXXXX
.......
XXXXXX.
...X.X.
.XXX.XX
...X...
```

file: maz3d.out

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%      A      M     M        A      ZZZZZ   333   DDDD   %
%     A A     MM   MM       A A        Z       3  D   D  %
%    AAAAA    M M M M      AAAAA      Z      33   D   D  %
%   A     A   M  M  M     A     A    Z         3  D   D  %
%  A       A  M     M    A       A  ZZZZZ   333   DDDD   %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Chris Schuster     MEMS/Robotics,Rice University     v6.03


Maze Environment...

  The Maze:   maz3d.dat    has    7 Rows ,  7 Cols ,  7 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  3,  3,  1)
        Goal  node = G , at (  3,  5,  7)

    Depth 1  Depth 2  Depth 3  Depth 4  Depth 5  Depth 6  Depth 7
    1234567  1234567  1234567  1234567  1234567  1234567  1234567

  1 ...X...  .XXX.X.  .X.X.X.  .X.XXXX  .X.X.X.  XXXX.X.  .....X.
  2 XX.XXXX  XXXXXXX  XX.X.X.  XXXXXXX  .X.X.X.  XXXXXXX  .XXXXXX
  3 ..SX...  .XXXXX.  .X.X.X.  XXXX.XX  ...X.X.  XXXXXXX  ....G..
  4 XXXX.XX  XXXXXXX  .X.XXXX  XXXXXXX  XXXXXX.  XXXXXXX  XXXXXX.
  5 .X.X...  .X.XXX.  .X.X...  XXXX.XX  ...X.X.  XX.X.X.  ...X.X.
  6 .X.XXXX  XXXXXXX  XXXXXXX  XXXXXXX  .XXXXXX  XXXXXXX  .XXX.XX
  7 .X.....  XXXXXX.  .......  .XXXXXX  .X.....  XX.XXX.  ...X...

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =   0.0000
  Connected obstacle value  (ObsC)  =   0.0000
  Start node value  - - - -  (Gnd)  =   0.0000
  Goal  node value  - - - -  (Vcc)  =  10.0000
  Max nodal change allowed per iter =   0.0010
  Max number of iterations allowed  =     3000
  Horiz directions allowed (4 or 8) =        4


Global minimal distance solution found using  104 iterations!!!
```

```
Final Nodal Potentials...


Depth Layer:  1
  0.193E+00  0.144E+00  0.962E-01  0.000E+00  0.714E+01  0.701E+01  0.687E+01
  0.000E+00  0.000E+00  0.480E-01  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.583E+01  0.595E+01  0.608E+01
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.570E+01  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.113E+01  0.000E+00  0.558E+01  0.545E+01  0.533E+01
  0.000E+00  0.000E+00  0.119E+01  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.125E+01  0.131E+01  0.137E+01  0.143E+01  0.149E+01

Depth Layer:  2
  0.241E+00  0.000E+00  0.000E+00  0.000E+00  0.728E+01  0.000E+00  0.674E+01
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.621E+01
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.107E+01  0.000E+00  0.000E+00  0.000E+00  0.521E+01
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.156E+01

Depth Layer:  3
  0.290E+00  0.000E+00  0.797E+00  0.000E+00  0.742E+01  0.000E+00  0.661E+01
  0.000E+00  0.000E+00  0.850E+00  0.000E+00  0.756E+01  0.000E+00  0.647E+01
  0.000E+00  0.000E+00  0.905E+00  0.000E+00  0.770E+01  0.000E+00  0.634E+01
  0.000E+00  0.000E+00  0.960E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.102E+01  0.000E+00  0.486E+01  0.498E+01  0.509E+01
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.204E+01  0.197E+01  0.190E+01  0.183E+01  0.176E+01  0.169E+01  0.162E+01

Depth Layer:  4
  0.339E+00  0.000E+00  0.744E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.784E+01  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.474E+01  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.211E+01  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00

Depth Layer:  5
  0.388E+00  0.000E+00  0.692E+00  0.000E+00  0.826E+01  0.000E+00  0.100E+02
  0.437E+00  0.000E+00  0.640E+00  0.000E+00  0.812E+01  0.000E+00  0.100E+02
  0.487E+00  0.538E+00  0.589E+00  0.000E+00  0.798E+01  0.000E+00  0.100E+02
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.100E+02
  0.234E+01  0.242E+01  0.250E+01  0.000E+00  0.463E+01  0.000E+00  0.100E+02
  0.226E+01  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.219E+01  0.000E+00  0.338E+01  0.348E+01  0.357E+01  0.367E+01  0.377E+01

Depth Layer:  6
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.841E+01  0.000E+00  0.100E+02
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.258E+01  0.000E+00  0.452E+01  0.000E+00  0.100E+02
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.000E+00  0.000E+00  0.329E+01  0.000E+00  0.000E+00  0.000E+00  0.388E+01

Depth Layer:  7
  0.913E+01  0.898E+01  0.884E+01  0.869E+01  0.855E+01  0.000E+00  0.100E+02
  0.927E+01  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
  0.942E+01  0.956E+01  0.971E+01  0.985E+01  0.100E+02  0.100E+02  0.100E+02
  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.100E+02
```

```
0.284E+01  0.275E+01  0.267E+01  0.000E+00  0.441E+01  0.000E+00  0.100E+02
0.292E+01  0.000E+00  0.000E+00  0.000E+00  0.430E+01  0.000E+00  0.000E+00
0.301E+01  0.310E+01  0.319E+01  0.000E+00  0.419E+01  0.408E+01  0.398E+01


             Total Iterations  =    3000
Maximum individual nodal change  =   0.0015
   Total iteration nodal change  =   0.0489


Solution Path...     Path = 104 steps,  104.00 units

  Step         Current Node        Next Move
 Number     ( Row, Col, Depth)     Direction

 START PT   (  3 ,  3 ,  1  )         N
     1      (  2 ,  3 ,  1  )         N
     2      (  1 ,  3 ,  1  )         W
     3      (  1 ,  2 ,  1  )         W
     4      (  1 ,  1 ,  1  )        DOWN
     5      (  1 ,  1 ,  2  )        DOWN
     6      (  1 ,  1 ,  3  )        DOWN
     7      (  1 ,  1 ,  4  )        DOWN
     8      (  1 ,  1 ,  5  )         S
     9      (  2 ,  1 ,  5  )         S
    10      (  3 ,  1 ,  5  )         E
    11      (  3 ,  2 ,  5  )         E
    12      (  3 ,  3 ,  5  )         N
    13      (  2 ,  3 ,  5  )         N
    14      (  1 ,  3 ,  5  )        UP
    15      (  1 ,  3 ,  4  )        UP
    16      (  1 ,  3 ,  3  )         S
    17      (  2 ,  3 ,  3  )         S
    18      (  3 ,  3 ,  3  )         S
    19      (  4 ,  3 ,  3  )         S
    20      (  5 ,  3 ,  3  )        UP
    21      (  5 ,  3 ,  2  )        UP
    22      (  5 ,  3 ,  1  )         S
    23      (  6 ,  3 ,  1  )         S
    24      (  7 ,  3 ,  1  )         E
    25      (  7 ,  4 ,  1  )         E
    26      (  7 ,  5 ,  1  )         E
    27      (  7 ,  6 ,  1  )         E
    28      (  7 ,  7 ,  1  )        DOWN
    29      (  7 ,  7 ,  2  )        DOWN
    30      (  7 ,  7 ,  3  )         W
    31      (  7 ,  6 ,  3  )         W
    32      (  7 ,  5 ,  3  )         W
    33      (  7 ,  4 ,  3  )         W
    34      (  7 ,  3 ,  3  )         W
    35      (  7 ,  2 ,  3  )         W
    36      (  7 ,  1 ,  3  )        DOWN
    37      (  7 ,  1 ,  4  )        DOWN
    38      (  7 ,  1 ,  5  )         N
    39      (  6 ,  1 ,  5  )         N
    40      (  5 ,  1 ,  5  )         E
    41      (  5 ,  2 ,  5  )         E
    42      (  5 ,  3 ,  5  )        DOWN
    43      (  5 ,  3 ,  6  )        DOWN
    44      (  5 ,  3 ,  7  )         W
    45      (  5 ,  2 ,  7  )         W
```

46	(5 , 1 , 7)	S
47	(6 , 1 , 7)	S
48	(7 , 1 , 7)	E
49	(7 , 2 , 7)	E
50	(7 , 3 , 7)	UP
51	(7 , 3 , 6)	UP
52	(7 , 3 , 5)	E
53	(7 , 4 , 5)	E
54	(7 , 5 , 5)	E
55	(7 , 6 , 5)	E
56	(7 , 7 , 5)	DOWN
57	(7 , 7 , 6)	DOWN
58	(7 , 7 , 7)	W
59	(7 , 6 , 7)	W
60	(7 , 5 , 7)	N
61	(6 , 5 , 7)	N
62	(5 , 5 , 7)	UP
63	(5 , 5 , 6)	UP
64	(5 , 5 , 5)	UP
65	(5 , 5 , 4)	UP
66	(5 , 5 , 3)	E
67	(5 , 6 , 3)	E
68	(5 , 7 , 3)	UP
69	(5 , 7 , 2)	UP
70	(5 , 7 , 1)	W
71	(5 , 6 , 1)	W
72	(5 , 5 , 1)	N
73	(4 , 5 , 1)	N
74	(3 , 5 , 1)	E
75	(3 , 6 , 1)	E
76	(3 , 7 , 1)	DOWN
77	(3 , 7 , 2)	DOWN
78	(3 , 7 , 3)	N
79	(2 , 7 , 3)	N
80	(1 , 7 , 3)	UP
81	(1 , 7 , 2)	UP
82	(1 , 7 , 1)	W
83	(1 , 6 , 1)	W
84	(1 , 5 , 1)	DOWN
85	(1 , 5 , 2)	DOWN
86	(1 , 5 , 3)	S
87	(2 , 5 , 3)	S
88	(3 , 5 , 3)	DOWN
89	(3 , 5 , 4)	DOWN
90	(3 , 5 , 5)	N
91	(2 , 5 , 5)	N
92	(1 , 5 , 5)	DOWN
93	(1 , 5 , 6)	DOWN
94	(1 , 5 , 7)	W
95	(1 , 4 , 7)	W
96	(1 , 3 , 7)	W
97	(1 , 2 , 7)	W
98	(1 , 1 , 7)	S
99	(2 , 1 , 7)	S
100	(3 , 1 , 7)	E
101	(3 , 2 , 7)	E
102	(3 , 3 , 7)	E
103	(3 , 4 , 7)	E
104	(3 , 5 , 7)	GOAL

```
Maze Solution...

  The Maze:   maz3d.dat    has   7 Rows ,  7 Cols ,  7 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  3,  3,  1)
        Goal  node = G , at (  3,  5,  7)

      Depth 1   Depth 2   Depth 3   Depth 4   Depth 5   Depth 6   Depth 7
      1234567   1234567   1234567   1234567   1234567   1234567   1234567

  1   432X432   5XXX5X1   6X6X6X0   7X5XXXX   8X4X2X.   XXXX3X.   87654X.
  2   XX1XXXX   XXXXXXX   XX7X7X9   XXXXXXX   9X3X1X.   XXXXXXX   9XXXXXX
  3   ..SX456   .XXXXX7   .X8X8X8   XXXX9XX   012X0X.   XXXXXXX   0123G..
  4   XXXX3XX   XXXXXXX   .X9XXXX   XXXXXXX   XXXXXX.   XXXXXXX   XXXXXX.
  5   .X2X210   .X1XXX9   .X0X678   XXXX5XX   012X4X.   XX3X3X.   654X2X.
  6   .X3XXXX   XXXXXXX   XXXXXXX   XXXXXXX   9XXXXXX   XXXXXXX   7XXX1XX
  7   .X45678   XXXXXX9   6543210   7XXXXXX   8X23456   XX1XXX7   890X098
```

file: mazno.out

```
Maze Environment....

  The Maze:   mazno.dat    has   7 Rows ,  7 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  7,   3,   1)
        Goal  node = G , at (  1,   7,   1)

      1234567

  1   X.....G
  2   ..X.XXX
  3   .X....X
  4   .XXXX..
  5   ....XX.
  6   .XX.X..
  7   ..S....

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =   0.0000
  Connected obstacle value  (ObsC)  =   0.0000
  Start node value  - - - -  (Gnd)  =   0.0000
  Goal  node value  - - - -  (Vcc)  =  10.0000
  Max nodal change allowed per iter =   0.0050
  Max number of iterations allowed  =     1000
  Horiz directions allowed (4 or 8) =        4


Global minimal distance solution found using   14 iterations!!!


Final Nodal Potentials...

  0.000E+00  0.501E+01  0.568E+01  0.635E+01  0.756E+01  0.878E+01  0.100E+02
  0.369E+01  0.435E+01  0.000E+00  0.580E+01  0.000E+00  0.000E+00  0.000E+00
  0.303E+01  0.000E+00  0.526E+01  0.527E+01  0.473E+01  0.421E+01  0.000E+00
  0.238E+01  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.369E+01  0.317E+01
  0.173E+01  0.152E+01  0.131E+01  0.110E+01  0.000E+00  0.000E+00  0.266E+01
  0.130E+01  0.000E+00  0.000E+00  0.892E+00  0.000E+00  0.191E+01  0.216E+01
  0.863E+00  0.431E+00  0.000E+00  0.688E+00  0.117E+01  0.166E+01  0.191E+01


               Total Iterations  =      226
Maximum individual nodal change  =   0.0050
   Total iteration nodal change  =   0.0431
```

```
Solution Path...     Path =  16 steps,    16.00 units

  Step          Current Node         Next Move
 Number      ( Row, Col, Depth)      Direction

 START PT    (  7 ,  3 ,  1   )          E
    1        (  7 ,  4 ,  1   )          E
    2        (  7 ,  5 ,  1   )          E
    3        (  7 ,  6 ,  1   )          N
    4        (  6 ,  6 ,  1   )          E
    5        (  6 ,  7 ,  1   )          N
    6        (  5 ,  7 ,  1   )          N
    7        (  4 ,  7 ,  1   )          W
    8        (  4 ,  6 ,  1   )          N
    9        (  3 ,  6 ,  1   )          W
   10        (  3 ,  5 ,  1   )          W
   11        (  3 ,  4 ,  1   )          N
   12        (  2 ,  4 ,  1   )          N
   13        (  1 ,  4 ,  1   )          E
   14        (  1 ,  5 ,  1   )          E
   15        (  1 ,  6 ,  1   )          E
   16        (  1 ,  7 ,  1   )         GOAL


Maze Solution...

  The Maze:   mazno.dat    has    7 Rows ,   7 Cols ,   1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,    @ (connected)
        Free Space = .    [ + D (doors) ,     E (elevatrrs) ]
        Start node = S , at (  7,   3,   1)
        Goal  node = G , at (  1,   7,   1)

      1234567

  1   X..345G
  2   ..X2XXX
  3   .X.109X
  4   .XXXX87
  5   ....XX6
  6   .XX.X45
  7   ..S123.
```

file: maznoc.out
(using iteration cut-off feature)

```
Maze Environment...

  The Maze:   mazno.dat    has   7 Rows ,  7 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  7,   3,   1)
        Goal  node = G , at (  1,   7,   1)

      1234567

  1   X.....G
  2   ..X.XXX
  3   .X....X
  4   .XXXX..
  5   ....XX.
  6   .XX.X..
  7   ..S....

Node Potential Parameters:

  Isolated  obstacle value   (ObsX)   =    .0000
  Connected obstacle value   (ObsC)   =    .0000
  Start node value  - - - -  (Gnd)    =    .0000
  Goal  node value  - - - -  (Vcc)    =  10.0000
  Max nodal change allowed per iter  =    .0100
  Max number of iterations allowed   =      100
  Horiz directions allowed (4 or 8)  =        4


Global minimal distance solution found using   14 iterations!!!


Final Nodal Potentials...

  0.000E+00  0.119E+01  0.176E+01  0.279E+01  0.497E+01  0.748E+01  0.100E+02
  0.380E+00  0.611E+00  0.000E+00  0.166E+01  0.000E+00  0.000E+00  0.000E+00
  0.149E+00  0.000E+00  0.761E+00  0.964E+00  0.475E+00  0.293E+00  0.000E+00
  0.820E-01  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.110E+00  0.627E-01
  0.152E-01  0.814E-02  0.109E-02  0.543E-03  0.000E+00  0.000E+00  0.156E-01
  0.814E-02  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.723E-03  0.567E-02
  0.109E-02  0.543E-03  0.000E+00  0.000E+00  0.000E+00  0.482E-03  0.723E-03


                Total Iterations  =       14
Maximum individual nodal change  =    .2301
   Total iteration nodal change  =   1.3488
```

```
Solution Path...     Path =  14 steps,   14.00 units

   Step          Current Node       Next Move
  Number      ( Row, Col, Depth)    Direction

 START PT     (  7 ,  3 ,  1   )        W
    1         (  7 ,  2 ,  1   )        W
    2         (  7 ,  1 ,  1   )        N
    3         (  6 ,  1 ,  1   )        N
    4         (  5 ,  1 ,  1   )        N
    5         (  4 ,  1 ,  1   )        N
    6         (  3 .  1 ,  1   )        N
    7         (  2 ,  1 ,  1   )        E
    8         (  2 ,  2 ,  1   )        N
    9         (  1 ,  2 ,  1   )        E
   10         (  1 ,  3 ,  1   )        E
   11         (  1 ,  4 ,  1   )        E
   12         (  1 ,  5 ,  1   )        E
   13         (  1 ,  6 ,  1   )        E
   14         (  1 ,  7 ,  1   )      GOAL

Maze Solution...

  The Maze:   mazno.dat    has   7 Rows ,  7 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,    @ (connected)
        Free Space = .    [ + D (doors) ,    E (elevators) ]
        Start node = S , at (  7,   3,   1)
        Goal  node = G , at (  1,   7,   1)

      1234567

  1   X90123G
  2   78X.XXX
  3   6X....X
  4   5XXXX..
  5   4...XX.
  6   3XX.X..
  7   21S....
```

file: mazn.out

```
Maze Environment...

  The Maze:   mazn.dat    has  15 Rows , 17 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  5, 14,  1)
        Goal  node = G , at (  8,  6,  1)

      12345678901234567

   1  XXXXXXXXXXXXXXXXX
   2  X......XX...X...X
   3  X......XX.X.X.X.X
   4  X.....XX..X.X.X.X
   5  X....XX..XX.XSX.X
   6  X...XX..XXX.XXX.X
   7  X...X..XXXX.X.X.X
   8  X...XGXX.XX.X.X.X
   9  X.........X...X.X
  10  X.........X.X.X.X
  11  X.........X.X.X.X
  12  X.........X.X.X.X
  13  X.........X.X.X.X
  14  X...........X...X
  15  XXXXXXXXXXXXXXXXX

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =   0.0000
  Connected obstacle value  (ObsC)  =   0.0000
  Start node value  - - - -  (Gnd)  =   0.0000
  Goal  node value  - - - -  (Vcc)  =  10.0000
  Max nodal change allowed per iter =   0.0010
  Max number of iterati ns allowed  =     5000
  Horiz directions allowed (4 or 8) =        4


Global minimal distance solution found using   43 iterations!!!


               Total Iterations  =    2037
Maximum individual nodal change  =  0.0010
   Total iteration nodal change  =  0.0444
```

```
Solution Path...     Path =  43 steps,    43.00 units

  Step        Current Node       Next Move
 Number     ( Row, Col, Depth)   Direction

 START PT   (  5 , 14 ,  1   )       N
    1       (  4 , 14 ,  1   )       N
    2       (  3 , 14 ,  1   )       N
    3       (  2 , 14 ,  1   )       E
    4       (  2 , 15 ,  1   )       E
    5       (  2 , 16 ,  1   )       S
    6       (  3 , 16 ,  1   )       S
    7       (  4 , 16 ,  1   )       S
    8       (  5 , 16 ,  1   )       S
    9       (  6 , 16 ,  1   )       S
   10       (  7 , 16 ,  1   )       S
   11       (  8 , 16 ,  1   )       S
   12       (  9 , 16 ,  1   )       S
   13       ( 10 , 16 ,  1   )       S
   14       ( 11 , 16 ,  1   )       S
   15       ( 12 , 16 ,  1   )       S
   16       ( 13 , 16 ,  1   )       S
   17       ( 14 , 1  ,  1   )       W
   18       ( 14 , 15 ,  1   )       W
   19       ( 14 , 14 ,  1   )       N
   20       ( 13 , 14 ,  1   )       N
   21       ( 12 , 14 ,  1   )       N
   22       ( 11 , 14 ,  1   )       N
   23       ( 10 , 14 ,  1   )       N
   24       (  9 , 14 ,  1   )       W
   25       (  9 , 13 ,  1   )       W
   26       (  9 , 12 ,  1   )       S
   27       ( 10 , 12 ,  1   )       S
   28       ( 11 , 12 ,  1   )       S
   29       ( 12 , 12 ,  1   )       S
   30       ( 13 , 12 ,  1   )       S
   31       ( 14 , 12 ,  1   )       W
   32       ( 14 , 11 ,  1   )       W
   33       ( 14 , 10 ,  1   )       N
   34       ( 13 , 10 ,  1   )       N
   35       ( 12 , 10 ,  1   )       N
   36       ( 11 , 10 ,  1   )       N
   37       ( 10 , 10 ,  1   )       N
   38       (  9 , 10 ,  1   )       W
   39       (  9 ,  9 ,  1   )       W
   40       (  9 ,  8 ,  1   )       W
   41       (  9 ,  7 ,  1   )       W
   42       (  9 ,  6 ,  1   )       N
   43       (  8 ,  6 ,  1   )      GOAL
```

```
Maze Solution...

  The Maze:   mazn.dat     has  15 Rows , 17 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  5, 14,  1)
        Goal  node = G , at (  8,  6,  1)

      12345678901234567

   1  XXXXXXXXXXXXXXXXX
   2  X......XX...X345X
   3  X......XX.X.X2X6X
   4  X.....XX..X.X1X7X
   5  X....XX..XX.XSX8X
   6  X...XX..XXX.XXX9X
   7  X...X..XXXX.X.X0X
   8  X...XGXX.XX.X.X1X
   9  X....21098X654X2X
  10  X........7X7X3X3X
  11  X........6X8X2X4X
  12  X........5X9X1X5X
  13  X........4X0X0X6X
  14  X........321X987X
  15  XXXXXXXXXXXXXXXXX
```

file: maznc.out (using iteration cut-off feature)

```
Maze Environment...

  The Maze:   mazn.dat     has  15 Rows , 17 Cols ,  1 Depth Layer(s)

  Key: Obstacles  = X (isolated) ,   @ (connected)
       Free Space = .   [ + D (doors) ,   E (elevators) ]
       Start node = S , at (  5, 14,  1)
       Goal  node = G , at (  8,  6,  1)

      12345678901234567

   1  XXXXXXXXXXXXXXXXX
   2  X......XX...X...X
   3  X......XX.X.X.X.X
   4  X.....XX..X.X.X.X
   5  X....XX..XX.XSX.X
   6  X...XX..XXX.XXX.X
   7  X...X..XXXX.X.X.X
   8  X...XGXX.XX.X.X.X
   9  X.........X...X.X
  10  X.........X.X.X.X
  11  X.........X.X.X.X
  12  X.........X.X.X.X
  13  X.........X.X.X.X
  14  X...........X...X
  15  XXXXXXXXXXXXXXXXX

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =    0.0000
  Connected obstacle value  (ObsC)  =    0.0000
  Start node value  - - - -  (Gnd)  =    0.0000
  Goal  node value  - - - -  (Vcc)  =   10.0000
  Max nodal change allowed per iter =    0.0100
  Max number of iterations allowed  =       100
  Horiz directions allowed (4 or 8) =         4


Global minimal distance solution found using   43 iterations!!!


               Total Iterations  =       43
Maximum individual nodal change  =   0.1126
   Total iteration nodal change  =   0.6078
```

```
Solution Path...    Path =  45 steps,   45.00 units

  Step          Current Node        Next Move
 Number      ( Row, Col, Depth)     Direction

START PT     (  5 , 14 ,   1   )       N
   1         (  4 , 14 ,   1   )       N
   2         (  3 , 14 ,   1   )       N
   3         (  2 , 14 ,   1   )       E
   4         (  2 , 15 ,   1   )       E
   5         (  2 , 16 ,   1   )       S
   6         (  3 , 16 ,   1   )       S
   7         (  4 , 16 ,   1   )       S
   8         (  5 , 16 ,   1   )       S
   9         (  6 , 16 ,   1   )       S
  10         (  7 , 16 ,   1   )       S
  11         (  8 , 16 ,   1   )       S
  12         (  9 , 16 ,   1   )       S
  13         ( 10 , 16 ,   1   )       S
  14         ( 11 , 16 ,   1   )       S
  15         ( 12 , 16 ,   1   )       S
  16         ( 13 , 16 ,   1   )       S
  17         ( 14 , 16 ,   1   )       W
  18         ( 14 , 15 ,   1   )       W
  19         ( 14 , 14 ,   1   )       N
  20         ( 13 , 14 ,   1   )       N
  21         ( 12 , 14 ,   1   )       N
  22         ( 11 , 14 ,   1   )       N
  23         ( 10 , 14 ,   1   )       N
  24         (  9 , 14 ,   1   )       W
  25         (  9 , 13 ,   1   )       W
  26         (  9 , 12 ,   1   )       N
  27         (  8 , 12 ,   1   )       N
  28         (  7 , 12 ,   1   )       N
  29         (  6 , 12 ,   1   )       N
  30         (  5 , 12 ,   1   )       N
  31         (  4 , 12 ,   1   )       N
  32         (  3 , 12 ,   1   )       N
  33         (  2 , 12 ,   1   )       W
  34         (  2 , 11 ,   1   )       W
  35         (  2 , 10 ,   1   )       S
  36         (  3 , 10 ,   1   )       S
  37         (  4 , 10 ,   1   )       W
  38         (  4 ,  9 ,   1   )       S
  39         (  5 ,  9 ,   1   )       W
  40         (  5 ,  8 ,   1   )       S
  41         (  6 ,  8 ,   1   )       W
  42         (  6 ,  7 ,   1   )       S
  43         (  7 ,  7 ,   1   )       W
  44         (  7 ,  6 ,   1   )       S
  45         (  8 ,  6 ,   1   )     GOAL
```

```
Maze Solution...

  The Maze:   mazn.dat     has  15 Rows , 17 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  5, 14,  1)
        Goal  node = G , at (  8,  6,  1)

        12345678901234567

   1    XXXXXXXXXXXXXXXXX
   2    X......XX543X345X
   3    X......XX6X2X2X6X
   4    X.....XX87X1X1X7X
   5    X....XX09XX0XSX8X
   6    X...XX21XXX9XXX9X
   7    X...X43XXXX8X.X0X
   8    X...XGXX.XX7X.X1X
   9    X.........X654X2X
  10    X.........X.X3X3X
  11    X.........X.X2X4X
  12    X.........X.X1X5X
  13    X.........X.X0X6X
  14    X...........X987X
  15    XXXXXXXXXXXXXXXXX
```

file: landnav.out

```
Maze Environment...

  The Maze:   landnav.dat  has  44 Rows , 64 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at ( 44, 32,  1)
        Goal  node = G , at (  4, 30,  1)

       1234567890123456789012345678901234567890123456789012345678901234

   1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
   2   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
   3   XXXXX.........XXXXXXXX...................X...XXX@@@@@......XXXXX
   4   XXXXX.........XXXXXXX........G...............XXX@@@@@......XXXXX
   5   XXXXXX........XXXXXXX........................XXX@@@@@.....XXXXXX
   6   XXXXXX........XXXXXXX........................XXX@@@@@.....XXXXXX
   7   XXXXXX........XXXXXXX........................XXX@@@@@....XXXXXXX
   8   XXXXXXX.......XXXXXXX........................XXX@@@@@....XXXXXXX
   9   XXXXXXXX......XXXXXXX@.......................XXX@@@@@....XXXXXXX
  10   XXXXXXXX......XXX@@@@@.......................XXX@@@@@...XXXXXXXX
  11   XXXXXXXXX.....XXXX@@@@........................XX@@@@@...XXXXXXXX
  12   XXXXXXXXX.....XXX@@@@@.......................XXX@@@@@..XXXXXXXXX
  13   XXXXXXXXX.....XXXX@@@@...........X...........XXX@@@@@..XXXXXXXXX
  14   XXXXXXXXXX....XXX@@@@@..........XXX..........XXX@@@@@.XXXXXXXXXX
  15   XXXXXXXXXX....XXXX@@@@.........XXXXX.........XXX@@@@@.XXXXXXXXXX
  16   XXXXXXXXXXX...XXX@@@@@.........XXXXX.........XXX@@@@@.XXXXXXXXXX
  17   XXXXXXXXXXX...XXXX@@@@.........XXXXX.........XXX@@@@@.XXXXXXXXXX
  18   XXXXXXXXXXX...XXX@@@@@..........@@@..........XXX@@@@@.XXXXXXXXXX
  19   XXXXXXXXXXXX..XXX@@@@@.........@@@@..........XXX@@@@@XXXXXXXXXXX
  20   XXXXXXXXXXXX..XXX@@@@@..........@@@..........XXX@@@@@XXXXXXXXXXX
  21   XXXXXXXXXXXXX.XXX@@@@@.......................XXX@@@@@XXXXXXXXXXX
  22   XXXXXXXXXXXXX.XXX@@@@@........................XX@@@@@XXXXXXXXXXX
  23   XXXXXXXXXXXXXXXXX@@@@@.......................XXX@@@XXXXXXXXXXXXX
  24   XXXXXXXXXXXXXXXXX@@@@@.....XXX...............XXX@@@XXXXXXXXXXXXX
  25   XXXXXXXXXXXXXXXXX@@@@@.....XXXXX.............XXX@@XXXXXXXXXXXXXX
  26   XXXXXXXXXXXXXXXXX@@@@@....XXXXXX.............XXX@@XXXXXXXXXXXXXX
  27   XXXXXXXXXXXXXXXXX@@@@@....XX@@XX.............XXX@@XXXXXXXXXXXXXX
  28   XXXXXXXXXXXXXXXXX@@@@@....XX@@@@.............XX@@@XXXXXXXXXXXXXX
  29   XXXXXXXXXXXXXXXXXX@@@@....X@@@@@.............XXX@@XXXXXXXXXXXXXX
  30   XXXXXXXXXXXXXXXXXX@@@@.....@@@@@.............XXX@@XXXXXXXXXXXXXX
  31   XXXXXXXXXXXXXXXXXX@@@@......@@@..............XXX@@XXXXXXXXXXXXXX
  32   XXXXXXXXXXXXXXXXXXX@@@........................XX@@XXXXXXXXXXXXXX
  33   XXXXXXXXXXXXXXXXXXX@@@........................XX@XXXXXXXXXXXXXXX
  34   XXXXXXXXXXXXXXXXXXX@@@.......................XXX@XXXXXXXXXXXXXXX
  35   XXXXXXXXXXXXXXXXXXXX@@........................XX@XXXXXXXXXXXXXXX
  36   XXXXXXXXXXXXXXXXXXXX@@.......................XXX@XXXXXXXXXXXXXXX
  37   XXXXXXXXXXXXXXXXXXXX@@.......................XXX@XXXXXXXXXXXXXXX
  38   XXXXXXXXXXXXXXXXXXXX@@.......................XX@XXXXXXXXXXXXXXXX
  39   XXXXXXXXXXXXXXXXXXXX@@........................XXXXXXXXXXXXXXXXXX
  40   XXXXXXXXXXXXXXXXXXXX@@.......................XXXXXXXXXXXXXXXXXXX
  41   XXXXXXXXXXXXXXXXXXX.@........................XXXXXXXXXXXXXXXXXXX
  42   XXXXXXXXXXXXXXXXXXX.....................@.X..XXXXXXXXXXXXXXXXXXX
  43   XXXXXXXXXXXXXXXXXXX.....................@@XX.XXXXXXXXXXXXXXXXXXX
  44   XXXXXXXXXXXXXXXXXXX............S.......@@@@@XXXXXXXXXXXXXXXXXXXX
```

```
Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =   0.0000
  Connected obstacle value  (ObsC)  =   0.0000
  Start node value  - - - -  (Gnd)   =   0.0000
  Goal  node value  - - - -  (Vcc)   =  10.0000
  Max nodal change allowed per iter =   0.0100
  Max number of iterations allowed  =      250
  Horiz directions allowed (4 or 8) =        8


Global minimal distance solution found using   40 iterations!!!


               Total Iterations  =      122
Maximum individual nodal change  =   0.0100
   Total iteration nodal change  =   2.4558


Solution Path...     Path =  40 steps,    46.63 units

   Step         Current Node          Next Move
  Number     ( Row, Col, Depth)       Direction

 START PT    ( 44 , 32 ,  1   )          N
     1       ( 43 , 32 ,  1   )          N
     2       ( 42 , 32 ,  1   )          N
     3       ( 41 , 32 ,  1   )          N
     4       ( 40 , 32 ,  1   )          N
     5       ( 39 , 32 ,  1   )         NE
     6       ( 38 , 33 ,  1   )         NE
     7       ( 37 , 34 ,  1   )         NE
     8       ( 36 , 35 ,  1   )          N
     9       ( 35 , 35 ,  1   )         NE
    10       ( 34 , 36 ,  1   )          N
    11       ( 33 , 36 ,  1   )          N
    12       ( 32 , 36 ,  1   )          N
    13       ( 31 , 36 ,  1   )          N
    14       ( 30 , 36 ,  1   )          N
    15       ( 29 , 36 ,  1   )          N
    16       ( 28 , 36 ,  1   )          N
    17       ( 27 , 36 ,  1   )          N
    18       ( 26 , 36 ,  1   )          N
    19       ( 25 , 36 ,  1   )          N
    20       ( 24 , 36 ,  1   )         NE
    21       ( 23 , 37 ,  1   )         NE
    22       ( 22 , 38 ,  1   )         NE
    23       ( 21 , 39 ,  1   )          N
    24       ( 20 , 39 ,  1   )          N
    25       ( 19 , 39 ,  1   )          N
    26       ( 18 , 39 ,  1   )          N
    27       ( 17 , 39 ,  1   )          N
    28       ( 16 , 39 ,  1   )         NW
    29       ( 15 , 38 ,  1   )         NW
    30       ( 14 , 37 ,  1   )         NW
    31       ( 13 , 36 ,  1   )         NW
    32       ( 12 , 35 ,  1   )         NW
    33       ( 11 , 34 ,  1   )         NW
    34       ( 10 , 33 ,  1   )         NW
    35       (  9 , 32 ,  1   )          N
```

```
36       ( 8 , 32 ,  1   )      NW
37       ( 7 , 31 ,  1   )       N
38       ( 6 , 31 ,  1   )       N
39       ( 5 , 31 ,  1   )      NW
40       ( 4 , 30 ,  1   )     GOAL


Maze Solution...

  The Maze:   landnav.dat  has  44 Rows , 64 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at ( 44, 32,  1)
        Goal  node = G , at (  4, 30,  1)

     1234567890123456789012345678901234567890123456789012345678901234

  1  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
  2  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
  3  XXXXX.........XXXXXXXX...................X...XXX@@@@@......XXXXX
  4  XXXXX.........XXXXXXX........G...............XXX@@@@@......XXXXX
  5  XXXXXX........XXXXXXX.........9..............XXX@@@@@.....XXXXXX
  6  XXXXXX........XXXXXXX.........8..............XXX@@@@@.....XXXXXX
  7  XXXXXX........XXXXXXX.........7..............XXX@@@@@....XXXXXXX
  8  XXXXXXX.......XXXXXXX..........6.............XXX@@@@@....XXXXXXX
  9  XXXXXXXX......XXXXXXX@.........5.............XXX@@@@@....XXXXXXX
 10  XXXXXXXX......XXX@@@@@..........4............XXX@@@@@...XXXXXXXX
 11  XXXXXXXXX.....XXXX@@@@...........3............XX@@@@@...XXXXXXXX
 12  XXXXXXXXX.....XXX@@@@@............2..........XXX@@@@@..XXXXXXXXX
 13  XXXXXXXXX.....XXXX@@@@...........X.1.........XXX@@@@@..XXXXXXXXX
 14  XXXXXXXXXX....XXX@@@@@..........XXX.0........XXX@@@@@.XXXXXXXXXX
 15  XXXXXXXXXX....XXXX@@@@.........XXXXX.9.......XXX@@@@@.XXXXXXXXXX
 16  XXXXXXXXXXX...XXX@@@@@.........XXXXX..8......XXX@@@@@.XXXXXXXXXX
 17  XXXXXXXXXXX...XXXX@@@@.........XXXXX..7......XXX@@@@@.XXXXXXXXXX
 18  XXXXXXXXXXX...XXX@@@@@..........@@@...6......XXX@@@@@.XXXXXXXXXX
 19  XXXXXXXXXXXX..XXX@@@@@.........@@@@...5......XXX@@@@@XXXXXXXXXXX
 20  XXXXXXXXXXXX..XXX@@@@@..........@@@...4......XXX@@@@@XXXXXXXXXXX
 21  XXXXXXXXXXXXX.XXX@@@@@................3......XXX@@@@@XXXXXXXXXXX
 22  XXXXXXXXXXXXX.XXX@@@@@...............2........XX@@@@@XXXXXXXXXXX
 23  XXXXXXXXXXXXXXXXX@@@@@..............1........XXX@@@@XXXXXXXXXXXX
 24  XXXXXXXXXXXXXXXXX@@@@@.....XXX.....0.........XXX@@@@XXXXXXXXXXXX
 25  XXXXXXXXXXXXXXXXX@@@@@.....XXXXX...9.........XXX@@@XXXXXXXXXXXXX
 26  XXXXXXXXXXXXXXXXX@@@@@....XXXXXX...8.........XXX@@@XXXXXXXXXXXXX
 27  XXXXXXXXXXXXXXXXX@@@@@....XX@@XX...7.........XXX@@@XXXXXXXXXXXXX
 28  XXXXXXXXXXXXXXXXX@@@@@....XX@@@@...6.........XX@@@XXXXXXXXXXXXXX
 29  XXXXXXXXXXXXXXXXXX@@@@....X@@@@@...5.........XXX@@XXXXXXXXXXXXXX
 30  XXXXXXXXXXXXXXXXXX@@@@.....@@@@@...4.........XXX@@XXXXXXXXXXXXXX
 31  XXXXXXXXXXXXXXXXXX@@@@......@@@....3.........XXX@@XXXXXXXXXXXXXX
 32  XXXXXXXXXXXXXXXXXXX@@@.............2..........XX@@XXXXXXXXXXXXXX
 33  XXXXXXXXXXXXXXXXXXX@@@.............1..........XX@XXXXXXXXXXXXXXX
 34  XXXXXXXXXXXXXXXXXXX@@@.............0.........XXX@XXXXXXXXXXXXXXX
 35  XXXXXXXXXXXXXXXXXXXX@@............9...........XX@XXXXXXXXXXXXXXX
 36  XXXXXXXXXXXXXXXXXXXX@@............8..........XXX@XXXXXXXXXXXXXXX
 37  XXXXXXXXXXXXXXXXXXXX@@...........7...........XXX@XXXXXXXXXXXXXXX
 38  XXXXXXXXXXXXXXXXXXXX@@..........6............XX@XXXXXXXXXXXXXXXX
 39  XXXXXXXXXXXXXXXXXXXX@@.........5..............XXXXXXXXXXXXXXXXXX
 40  XXXXXXXXXXXXXXXXXXXX@@.........4.............XXXXXXXXXXXXXXXXXXX
 41  XXXXXXXXXXXXXXXXXXX.@..........3.............XXXXXXXXXXXXXXXXXXX
 42  XXXXXXXXXXXXXXXXXXX............2........@.X..XXXXXXXXXXXXXXXXXXX
 43  XXXXXXXXXXXXXXXXXXX............1........@@XX.XXXXXXXXXXXXXXXXXXX
 44  XXXXXXXXXXXXXXXXXXX............S.......@@@@@XXXXXXXXXXXXXXXXXXXX
```

file: landnavc.out (using iteration cut-off feature)

```
Maze Environment...

  The Maze:   landnav.dat    has  44 Rows , 64 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) .]
        Start node = S , at ( 44, 32,  1)
        Goal  node = G , at (  4, 30,  1)

        1234567890123456789012345678901234567890123456789012345678901234

   1    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
   2    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
   3    XXXXX.........XXXXXXXX...................X...XXX@@@@@......XXXXX
   4    XXXXX.........XXXXXXX........G...............XXX@@@@@......XXXXX
   5    XXXXXX........XXXXXXX........................XXX@@@@@.....XXXXXX
   6    XXXXXX........XXXXXXX........................XXX@@@@@.....XXXXXX
   7    XXXXXX........XXXXXXX........................XXX@@@@@....XXXXXXX
   8    XXXXXXX.......XXXXXXX........................XXX@@@@@....XXXXXXX
   9    XXXXXXXX......XXXXXXX@.......................XXX@@@@@....XXXXXXX
  10    XXXXXXXX......XXX@@@@@.......................XXX@@@@@...XXXXXXXX
  11    XXXXXXXXX.....XXXX@@@@........................XX@@@@@...XXXXXXXX
  12    XXXXXXXXX.....XXX@@@@@.......................XXX@@@@@..XXXXXXXXX
  13    XXXXXXXXX.....XXXX@@@@...........X...........XXX@@@@@..XXXXXXXXX
  14    XXXXXXXXXX....XXX@@@@@..........XXX..........XXX@@@@@.XXXXXXXXXX
  15    XXXXXXXXXX....XXXX@@@@.........XXXXX.........XXX@@@@@.XXXXXXXXXX
  16    XXXXXXXXXXX...XXX@@@@@.........XXXXX.........XXX@@@@@.XXXXXXXXXX
  17    XXXXXXXXXXX...XXXX@@@@.........XXXXX.........XXX@@@@@.XXXXXXXXXX
  18    XXXXXXXXXXX...XXX@@@@@..........@@@..........XXX@@@@@.XXXXXXXXXX
  19    XXXXXXXXXXXX..XXX@@@@@.........@@@@..........XXX@@@@@XXXXXXXXXXX
  20    XXXXXXXXXXXX..XXX@@@@@..........@@@..........XXX@@@@@XXXXXXXXXXX
  21    XXXXXXXXXXXXX.XXX@@@@@.......................XXX@@@@@XXXXXXXXXXX
  22    XXXXXXXXXXXXX.XXX@@@@@........................XX@@@@@XXXXXXXXXXX
  23    XXXXXXXXXXXXXXXXX@@@@@.......................XXX@@@XXXXXXXXXXXXX
  24    XXXXXXXXXXXXXXXXX@@@@@.....XXX...............XXX@@@XXXXXXXXXXXXX
  25    XXXXXXXXXXXXXXXXX@@@@@.....XXXXX.............XXX@@XXXXXXXXXXXXXX
  26    XXXXXXXXXXXXXXXXX@@@@@....XXXXXX.............XXX@@XXXXXXXXXXXXXX
  27    XXXXXXXXXXXXXXXXX@@@@@....XX@@XX.............XXX@@XXXXXXXXXXXXXX
  28    XXXXXXXXXXXXXXXXX@@@@@....XX@@@@.............XX@@XXXXXXXXXXXXXXX
  29    XXXXXXXXXXXXXXXXXX@@@@....X@@@@@.............XXX@@XXXXXXXXXXXXXX
  30    XXXXXXXXXXXXXXXXXX@@@@.....@@@@@.............XXX@@XXXXXXXXXXXXXX
  31    XXXXXXXXXXXXXXXXXX@@@@......@@@..............XXX@@XXXXXXXXXXXXXX
  32    XXXXXXXXXXXXXXXXXXX@@@........................XX@@XXXXXXXXXXXXXX
  33    XXXXXXXXXXXXXXXXXXX@@@........................XX@XXXXXXXXXXXXXXX
  34    XXXXXXXXXXXXXXXXXXX@@@.......................XXX@XXXXXXXXXXXXXXX
  35    XXXXXXXXXXXXXXXXXXXX@@........................XX@XXXXXXXXXXXXXXX
  36    XXXXXXXXXXXXXXXXXXXX@@.......................XXX@XXXXXXXXXXXXXXX
  37    XXXXXXXXXXXXXXXXXXXX@@.......................XXX@XXXXXXXXXXXXXXX
  38    XXXXXXXXXXXXXXXXXXXX@@.......................XX@XXXXXXXXXXXXXXXX
  39    XXXXXXXXXXXXXXXXXXXX@@........................XXXXXXXXXXXXXXXXXX
  40    XXXXXXXXXXXXXXXXXXXX@@.......................XXXXXXXXXXXXXXXXXXX
  41    XXXXXXXXXXXXXXXXXXX.@........................XXXXXXXXXXXXXXXXXXX
  42    XXXXXXXXXXXXXXXXXXX.....................@.X..XXXXXXXXXXXXXXXXXXX
  43    XXXXXXXXXXXXXXXXXXX.....................@@XX.XXXXXXXXXXXXXXXXXXX
  44    XXXXXXXXXXXXXXXXXXX............S.......@@@@@XXXXXXXXXXXXXXXXXXXX
```

```
Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =    .0000
  Connected obstacle value  (ObsC)  =    .0000
  Start node value  - - - -  (Gnd)  =    .0000
  Goal  node value  - - - -  (Vcc)  =  10.0000
  Max nodal change allowed per iter =    .0100
  Max number of iterations allowed  =      200
  Horiz directions allowed (4 or 8) =        8


Global minimal distance solution found using   40 iterations!!!


               Total Iterations  =       40
Maximum individual nodal change  =    .0354
   Total iteration nodal change  =   4.3567


Solution Path...     Path =  40 steps,    44.14 units

   Step       Current Node        Next Move
  Number    ( Row, Col, Depth)    Direction

 START PT   ( 44 , 32 ,  1   )        N
    1       ( 43 , 32 ,  1   )        N
    2       ( 42 , 32 ,  1   )        N
    3       ( 41 , 32 ,  1   )        N
    4       ( 40 , 32 ,  1   )        N
    5       ( 39 , 32 ,  1   )        N
    6       ( 38 , 32 ,  1   )        N
    7       ( 37 , 32 ,  1   )        N
    8       ( 36 , 32 ,  1   )       NE
    9       ( 35 , 33 ,  1   )        N
   10       ( 34 , 33 ,  1   )       NE
   11       ( 33 , 34 ,  1   )        N
   12       ( 32 , 34 ,  1   )        N
   13       ( 31 , 34 ,  1   )        N
   14       ( 30 , 34 ,  1   )        N
   15       ( 29 , 34 ,  1   )        N
   16       ( 28 , 34 ,  1   )        N
   17       ( 27 , 34 ,  1   )       NW
   18       ( 26 , 33 ,  1   )        N
   19       ( 25 , 33 ,  1   )       NW
   20       ( 24 , 32 ,  1   )       NW
   21       ( 23 , 31 ,  1   )       NW
   22       ( 22 , 30 ,  1   )        N
   23       ( 21 , 30 ,  1   )        N
   24       ( 20 , 30 ,  1   )        N
   25       ( 19 , 30 ,  1   )        N
   26       ( 18 , 30 ,  1   )        N
   27       ( 17 , 30 ,  1   )        N
   28       ( 16 , 30 ,  1   )       NE
   29       ( 15 , 31 ,  1   )       NE
   30       ( 14 , 32 ,  1   )        N
   31       ( 13 , 32 ,  1   )        N
   32       ( 12 , 32 ,  1   )        N
   33       ( 11 , 32 ,  1   )        N
   34       ( 10 , 32 ,  1   )        N
   35       (  9 , 32 ,  1   )        N
 36       (  8 , 32 ,  1   )       NW
 37       (  7 , 31 ,  1   )        N
 38       (  6 , 31 ,  1   )        N
 39       (  5 , 31 ,  1   )       NW
 40       (  4 , 30 ,  1   )      GOAL
```

```
Maze Solution...

 The Maze:   landnav.dat     has  44 Rows , 64 Cols ,  1 Depth Layer(s)

 Key:  Obstacles  = X (isolated) ,   @ (connected)
       Free Space = .   [ + D (doors) ,   E (elevators) ]
       Start node = S , at ( 44, 32,  1)
       Goal  node = G , at (  4, 30,  1)

     1234567890123456789012345678901234567890123456789012345678901234

  1  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
  2  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX@@@@@.XXXXXXXXXX
  3  XXXXX.........XXXXXXXX...................X...XXX@@@@@......XXXXX
  4  XXXXX.........XXXXXXX........G...............XXX@@@@@......XXXXX
  5  XXXXXX........XXXXXXX.........9..............XXX@@@@@.....XXXXXX
  6  XXXXXX........XXXXXXX.........8..............XXX@@@@@.....XXXXXX
  7  XXXXXX........XXXXXXX.........7..............XXX@@@@@....XXXXXXX
  8  XXXXXXX.......XXXXXXX..........6.............XXX@@@@@....XXXXXXX
  9  XXXXXXXX......XXXXXXX@.........5.............XXX@@@@@....XXXXXXX
 10  XXXXXXXX......XXX@@@@@.........4.............XXX@@@@@...XXXXXXXX
 11  XXXXXXXXX.....XXXX@@@@.........3..............XX@@@@@...XXXXXXXX
 12  XXXXXXXXX.....XXX@@@@@.........2.............XXX@@@@@..XXXXXXXXX
 13  XXXXXXXXX.....XXXX@@@@.........1.X...........XXX@@@@@..XXXXXXXXX
 14  XXXXXXXXXX....XXX@@@@@.........0XXX..........XXX@@@@@.XXXXXXXXXX
 15  XXXXXXXXXX....XXXX@@@@........9XXXXX.........XXX@@@@@.XXXXXXXXXX
 16  XXXXXXXXXXX...XXX@@@@@.......8.XXXXX.........XXX@@@@@.XXXXXXXXXX
 17  XXXXXXXXXXX...XXXX@@@@.......7.XXXXX.........XXX@@@@@.XXXXXXXXXX
 18  XXXXXXXXXXX...XXX@@@@@.......6..@@@..........XXX@@@@@.XXXXXXXXXX
 19  XXXXXXXXXXXX..XXX@@@@@.......5.@@@@..........XXX@@@@@XXXXXXXXXXX
 20  XXXXXXXXXXXX..XXX@@@@@.......4..@@@..........XXX@@@@@XXXXXXXXXXX
 21  XXXXXXXXXXXXX.XXX@@@@@.......3...............XXX@@@@@XXXXXXXXXXX
 22  XXXXXXXXXXXXX.XXX@@@@@.......2................XX@@@@@XXXXXXXXXXX
 23  XXXXXXXXXXXXXXXXX@@@@@........1..............XXX@@@@XXXXXXXXXXXX
 24  XXXXXXXXXXXXXXXXX@@@@@.....XXX.0.............XXX@@@@XXXXXXXXXXXX
 25  XXXXXXXXXXXXXXXXX@@@@@.....XXXXX9............XXX@@XXXXXXXXXXXXXX
 26  XXXXXXXXXXXXXXXXX@@@@@....XXXXXX8............XXX@@XXXXXXXXXXXXXX
 27  XXXXXXXXXXXXXXXXX@@@@@....XX@@XX.7...........XXX@@XXXXXXXXXXXXXX
 28  XXXXXXXXXXXXXXXXX@@@@@....XX@@@@.6...........XX@@@XXXXXXXXXXXXXX
 29  XXXXXXXXXXXXXXXXXX@@@@....X@@@@@.5...........XXX@@XXXXXXXXXXXXXX
 30  XXXXXXXXXXXXXXXXXX@@@@.....@@@@@.4...........XXX@@XXXXXXXXXXXXXX
 31  XXXXXXXXXXXXXXXXXX@@@@......@@@..3...........XXX@@XXXXXXXXXXXXXX
 32  XXXXXXXXXXXXXXXXXXX@@@...........2............XX@@XXXXXXXXXXXXXX
 33  XXXXXXXXXXXXXXXXXXX@@@...........1............XX@XXXXXXXXXXXXXXX
 34  XXXXXXXXXXXXXXXXXXX@@@..........0............XXX@XXXXXXXXXXXXXXX
 35  XXXXXXXXXXXXXXXXXXXX@@..........9.............XX@XXXXXXXXXXXXXXX
 36  XXXXXXXXXXXXXXXXXXXX@@.........8.............XXX@XXXXXXXXXXXXXXX
 37  XXXXXXXXXXXXXXXXXXXX@@.........7.............XXX@XXXXXXXXXXXXXXX
 38  XXXXXXXXXXXXXXXXXXXX@@.........6.............XX@XXXXXXXXXXXXXXXX
 39  XXXXXXXXXXXXXXXXXXXX@@.........5..............XXXXXXXXXXXXXXXXXX
 40  XXXXXXXXXXXXXXXXXXXX@@.........4.............XXXXXXXXXXXXXXXXXXX
 41  XXXXXXXXXXXXXXXXXXX.@..........3.............XXXXXXXXXXXXXXXXXXX
 42  XXXXXXXXXXXXXXXXXXX............2........@.X..XXXXXXXXXXXXXXXXXXX
 43  XXXXXXXXXXXXXXXXXXX............1........@@XX.XXXXXXXXXXXXXXXXXXX
 44  XXXXXXXXXXXXXXXXXXX............S.......@@@@@XXXXXXXXXXXXXXXXXXXX
```

file: bldgnav.out

```
Maze Environment...

  The Maze:   bldgnav.dat  has  30 Rows , 80 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,    @ (connected)
        Free Space = .    [ + D (doors) ,    E (elevators) ]
        Start node = S , at (  3,  3,  1)
        Goal  node = G , at ( 24, 12,  1)

    12345678901234567890123456789012345678901234567890123456789012345678901234567890

 1  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2  X........X...................X.........X...........X.................X.........X
 3  X.S......X...................X.........X...........X.................X.........X
 4  X........X...................X.........X...........D.................X.........X
 5  X........X...................X.........X...........X.................X.........X
 6  X........X...................X.........X...........X.................X.........X
 7  XXXXXXXDXXXXXXXXXXXXXXXXXDXXXXXXXXDXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXX
 8  X..............................................................................X
 9  X..............................................................................X
10  X..............................................................................X
11  X....XXXXXXXXXXXXXXXXXXXXXXXXX.........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
12  X....D..........XXX...X......X.........X........X...............X....X.........X
13  X....X..........X.....D......X.........X...XX...X..........XX...X....X.....XX..X
14  X....XX...XXXXXXX.....X......X.........X...XX...X..........XX...X....X.....XX..X
15  XXXXXXX...D.....X.....XXXXXXXX.........D...XX...X..........XX...X....D.....XX..X
16  X....D....X.....X.....X......X.........X........X..........XX...X....X.........X
17  XXX..X....X.....X.....X......X.........X........X....XXXX.......X....X.........X
18  X....X....X.....X.....X......X.........X........X...............X....X....XX...X
19  XXXXXXXXXXXXXXXXXXXDXXXXXXXDXX.........XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
20  X..............................................................................X
21  X..............................................................................X
22  X..............................................................................X
23  XXXXXXXDXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXXXXXXXXXXDXXXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
24  X........X.G..X.....X........X.........X...........X.......XX........XXX.......X
25  X........X....X.....X........X.........X...........X.......XX........XXX.......X
26  X........X....X.....X........X.........X...........X.......XX........XXX.......X
27  X..X..X..X....D.....X........X.........X..XX.......D.......XX........X.........X
28  X..XXXX..X....X..............X...XXXX..X..XX.......X...XXXXXX......XXX....XXXX.X
29  X........X....X..........XXX.X.........X..XX.......X...XXXXXX......XXX....XXXX.X
30  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =    .0000
  Connected obstacle value  (ObsC)  =    .0000
  Start node value  - - - -  (Gnd)  =    .0000
  Goal  node value  - - - -  (Vcc)  =  10.0000
  Max nodal change allowed per iter =    .0100
  Max number of iterations allowed  =      200
  Horiz directions allowed (4 or 8) =        8


Global minimal distance solution found using   55 iterations!!!


               Total Iterations  =      200
```

```
Maximum individual nodal change  =    .0118
   Total iteration nodal change  =    .8483


Solution Path...     Path =  57 steps,    64.87 units

  Step        Current Node          Next Move
 Number    ( Row, Col, Depth)       Direction

 START PT   (  3 ,  3 ,  1   )          E
     1      (  3 ,  4 ,  1   )         SE
     2      (  4 ,  5 ,  1   )         SE
     3      (  5 ,  6 ,  1   )         SE
     4      (  6 ,  7 ,  1   )         SE
     5      (  7 ,  8 ,  1   )         SE
     6      (  8 ,  9 ,  1   )          E
     7      (  8 , 10 ,  1   )          E
     8      (  8 , 11 ,  1   )          E
     9      (  8   12 ,  1   )          E
    10      (  8 , 13 ,  1   )          E
    11      (  8 , 14 ,  1   )          E
    12      (  8 , 15 ,  1   )          E
    13      (  8 , 16 ,  1   )          E
    14      (  8 , 17 ,  1   )          E
    15      (  8 , 18 ,  1   )          E
    16      (  8 , 19 ,  1   )          E
    17      (  8 , 20 ,  1   )          E
    18      (  8 , 21 ,  1   )          E
    19      (  8 , 22 ,  1   )          E
    20      (  8 , 23 ,  1   )          E
    21      (  8 , 24 ,  1   )          E
    22      (  8 , 25 ,  1   )         SE
    23      (  9 , 26 ,  1   )          E
    24      (  9 , 27 ,  1   )          E
    25      (  9 , 28 ,  1   )          E
    26      (  9 , 29 ,  1   )         SE
    27      ( 10 , 30 ,  1   )         SE
    28      ( 11 , 31 ,  1   )          S
    29      ( 12 , 31 ,  1   )          S
    30      ( 13 , 31 ,  1   )          S
    31      ( 14 , 31 ,  1   )          S
    32      ( 15 , 31 ,  1   )          S
    33      ( 16 , 31 ,  1   )          S
    34      ( 17 , 31 ,  1   )          S
    35      ( 18 , 31 ,  1   )          S
    36      ( 19 , 31 ,  1   )         SW
    37      ( 20 , 30 ,  1   )         SW
    38      ( 21 , 29 ,  1   )         SW
    39      ( 22 , 28 ,  1   )         SW
    40      ( 23 , 27 ,  1   )          S
    41      ( 24 , 27 ,  1   )         SW
    42      ( 25 , 26 ,  1   )         SW
    43      ( 26 , 25 ,  1   )         SW
    44      ( 27 , 24 ,  1   )         SW
    45      ( 28 , 23 ,  1   )          W
    46      ( 28 , 22 ,  1   )          W
    47      ( 28 , 21 ,  1   )          W
    48      ( 28 , 20 ,  1   )          W
    49      ( 28 , 19 ,  1   )          W
    50      ( 28 , 18 ,  1   )          W
    51      ( 28 , 17 ,  1   )          W
   52       ( 28 , 16 ,  1   )       NW
   53       ( 27 , 15 ,  1   )        W
   54       ( 27 , 14 ,  1   )       NW
   55       ( 26 , 13 ,  1   )       NW
   56       ( 25 , 12 ,  1   )        N
   57       ( 24 , 12 ,  1   )      GOAL


Maze Solution...

  The Maze:   bldgnav.dat  has  30 Rows , 80 Cols ,  1 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,    @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  3,  3,  1)
        Goal  node = G , at ( 24, 12,  1)

     12345678901234567890123456789012345678901234567890123456789012345678901234567890

  1  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2  X........X...................X.........X...........X.................X.........X
  3  X.S1.....X...................X.........X...........X.................X.........X
  4  X...2....X...................X.........X...........D.................X.........X
  5  X....3.......................X.........X...........X.................X.........X
  6  X.....4..X...................X.........X...........X.................X.........X
  7  XXXXXXX5XXXXXXXXXXXXXXXXXDXXXXXXXXDXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXX
  8  X.......67890123456789012......................................................X
  9  X........................3456..................................................X
 10  X............................7.................................................X
 11  X....XXXXXXXXXXXXXXXXXXXXXXXXX8........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 12  X....D..........XXX...X......X9........X........X...............X....X.........X
 13  X....X..........X.....D......X0........X...XX...X..........XX...X....X.....XX..X
 14  X....XX...XXXXXXX.....X......X1........X...XX...X..........XX...X....X.....XX..X
 15  XXXXXXX...D.....X.....XXXXXXXX2........D...XX...X..........XX...X....D.....XX..X
 16  X....D....X.....X.....X......X3........X........X..........XX...X....X.........X
 17  XXX..X....X.....X.....X......X4........X........X....XXXX.......X....X.........X
 18  X....X....X.....X.....X......X5........X........X...............X....X....XX...X
 19  XXXXXXXXXXXXXXXXXXXDXXXXXXXDXX6........XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
 20  X............................7.................................................X
 21  X...........................8..................................................X
 22  X..........................9...................................................X
 23  XXXXXXXDXXXXXXXXXXXXXXXXXX0XXXXXXXDXXXXXXXXXXXXXD.XXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
 24  X........X.G..X.....X.....1..X.........X...........X.......XX... ....XXX.......X
 25  X........X.6..X.....X....2...X.........X...........X.......XX........XXX.......X
 26  X........X..5.X.....X...3....X.........X...........X.......XX........XXX.......X
 27  X..X..X..X...43.....X..4.....X.........X..XX..... .D.......XX........X.........X
 28  X..XXXX..X....X21098765......X...XXXX..X..XX.......X...XXXXXX......XXX....XXXX.X
 29  X........X....X..........XXX.X.........X..XX.......X...XXXXXX......XXX....XXXX.X
 30  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

file: b3dnav.out

```
Maze Environment...

  The Maze:   b3dnav.dat   has  30 Rows , 80 Cols ,  3 Depth Layer(s)

  Key:  Obstacles  = X (isolated) ,   @ (connected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  3,  3,  1)
        Goal  node = G , at ( 17,  3,  3)

      12345678901234567890123456789012345678901234567890123456789012345678901234567890


Depth Layer:  1
  1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2   X........X...................X.........X...........X.................X.........X
  3   X.S......X...................X.........X...........X.................X.........X
  4   X........X...................X.........X...........D.................X.........X
  5   X........X...................X.........X...........X.................X.........X
  6   X........X...................X.........X...........X.................X.........X
  7   XXXXXXXDXXXXXXXXXXXXXXXXXDXXXXXXXXDXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXXDXXXXXXXEXXXXX
  8   X................ .............................................................X
  9   X..............................................................................X
 10   X..............................................................................X
 11   X....XXXXXXXXXXXXXXXXXXXXXXXXX.........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 12   X....D..........XXX...X......X.........X........X...............X....X.........X
 13   X....X..........X.....D......X.........X...XX...X..........XX...X....X.....XX..X
 14   X....XX...XXXXXXX.....X......X.........X...XX...X..........XX...X....X.....XX..X
 15   XXXXXXX...D.....X.....XXXXXXXX.........D...XX...X..........XX...X....D.....XX..X
 16   X....D....X.....X.....X......X.........X........X..........XX...X....X.........X
 17   XXX..X....X.....X.....X......X.........X........X....XXXX.......X....X.........X
 18   X....X....X.....X.....X......X.........X........X...............X....X....XX...X
 19   XXXXXXXXXXXXXXXXXXXDXXXXXXXDXX.........XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
 20   X........................................... ..................................X
 21   X..............................................................................X
 22   X..............................................................................X
 23   XXXXXXXDXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXXXXXXXXXXDXXXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
 24   X........X....X.....X........X.........X...........X.......XX........XXX.......X
 25   X........X....X.....X........X.........X...........X.......XX........XXX.......X
 26   X........X....X.....X........X.........X...........X.......XX........XXX.......X
 27   X..X..X..X....D.....X........X.........X..XX.......D.......XX........X.........X
 28   X..XXXX. X....X..............X...XXXX..X..XX.......X...XXXXXX......XXX....XXXX.X
 29   X.... ...X....X..........XXX.X.........X..XX.......X...XXXXXX......XXX....XXXX.X
 30   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Depth Layer:  2
  1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  5   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  6   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  7   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXEXXXXX
  8   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  9   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 11   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 13   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 14    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 15    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 16    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 17    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 19    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 20    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 21    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 22    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 23    XXXXXXXXXXXXXXXX.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 24    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 25    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 26    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 27    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 28    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 29    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 30    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Depth Layer:  3
  1    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2    X........X...................X.........X...........X.................X.........X
  3    X........X...................X.........X...........X.................X.........X
  4    X........X...................X.........X...........D.................X.........X
  5    X........X...................X.........X...........X.................X.........X
  6    X........X...................X.........X...........X.................X.........X
  7    XXXXXXXDXXXXXXXXXXXXXXXXXDXXXXXXXXDXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXXXDXXXXXXEXXXXX
  8    X..............................................................................X
  9    X..............................................................................X
 10    X..............................................................................X
 11    XXXXXXXXXXXXXXXXXXXXXXXDXXXXXX.........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 12    X....X........X..............X.........X........X...............X....X.........X
 13    X....D........X..............X.........D........X...............X....D.........X
 14    X....X........X..............X.........X........X...............X....X.........X
 15    XXXXXX........XXXXXXXXXXXXXXXX.........XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
 16    X....X........X.......X......X.........X........X...............X....X.........X
 17    X.G..D........X.......X......X.........D........X...............X....D.........X
 18    X....X........X.......X......X.........X........X...............X....X.........X
 19    XXXXXXXXXDXXXXXXXXXDXXXXXXDXXX.........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 20    X..............................................................................X
 21    X..............................................................................X
 22    X..............................................................................X
 23    XXXXXXXDXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXXXXXXXXXXDXXXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
 24    X........X....X.....X........X.........X...........X.......XX........XXX.......X
 25    X........X....X.....X........X.........X...........X.......XX........XXX.......X
 26    X........X....X.....X........X.........X...........X.......XX........XXX.......X
 27    X........X... D.....X........X.........X..XX.......D.......XX........X.........X
 28    X........X....X......... ....X.........X..XX.......X...XXXXXX......XXX...XXXX.X
 29    X........X....X..............X.........X..XX.......X...XXXXXX......XXX...XXXX.X
 30    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Node Potential Parameters:

  Isolated  obstacle value  (ObsX)  =   0.0000
  Connected obstacle value  (ObsC)  =   0.0000
  Start node value  - - - -  (Gnd)  =   0.0000
  Goal  node value  - - - -  (Vcc)  =  10.0000
  Max nodal change allowed per iter =   0.0100
  Max number of iterations allowed  =      200
  Horiz directions allowed (4 or 8) =        8
```

```
Global minimal distance solution found using  196 iterations!!!


                 Total Iterations  =      200
Maximum individual nodal change  =   0.0102
   Total iteration nodal change  =   0.9461



Solution Path...     Path = 149 steps,  156.87 units

  Step          Current Node         Next Move
 Number      ( Row, Col, Depth)      Direction

 START PT    (  3 ,  3 ,  1   )         SE
    1        (  4 ,  4 ,  1   )         SE
    2        (  5 ,  5 ,  1   )         SE
    3        (  6 ,  6 ,  1   )          E
    4        (  6 ,  7 ,  1   )         SE
    5        (  7 ,  8 ,  1   )         SE
    6        (  8 ,  9 ,  1   )          E
    7        (  8 , 10 ,  1   )          E
    8        (  8 , 11 ,  1   )          E
    9        (  8 , 12 ,  1   )          E
   10        (  8 , 13 ,  1   )          E
   11        (  8 , 14 ,  1   )          E
   12        (  8 , 15 ,  1   )          E
   13        (  8 , 16 ,  1   )          E
   14        (  8 , 17 ,  1   )          E
   15        (  8 , 18 ,  1   )          E
   16        (  8 , 19 ,  1   )          E
   17        (  8 , 20 ,  1   )          E
   18        (  8 , 21 ,  1   )          E
   19        (  8 , 22 ,  1   )          E
   20        (  8 , 23 ,  1   )          E
   21        (  8 , 24 ,  1   )          E
   22        (  8 , 25 ,  1   )          E
   23        (  8 , 26 ,  1   )          E
   24        (  8 , 27 ,  1   )          E
   25        (  8 , 28 ,  1   )          E
   26        (  8 , 29 ,  1   )          E
   27        (  8 , 30 ,  1   )          E
   28        (  8 , 31 ,  1   )          E
   29        (  8 , 32 ,  1   )          E
   30        (  8 , 33 ,  1   )          E
   31        (  8 , 34 ,  1   )          E
   32        (  8 , 35 ,  1   )          E
   33        (  8 , 36 ,  1   )          E
   34        (  8 , 37 ,  1   )          E
   35        (  8 , 38 ,  1   )          E
   36        (  8 , 39 ,  1   )          E
   37        (  8 , 40 ,  1   )          E
   38        (  8 , 41 ,  1   )          E
   39        (  8 , 42 ,  1   )          E
   40        (  8 , 43 ,  1   )          E
   41        (  8 , 44 ,  1   )          E
   42        (  8 , 45 ,  1   )          E
   43        (  8 , 46 ,  1   )          E
   44        (  8 , 47 ,  1   )          E
   45        (  8 , 48 ,  1   )          E
   46        (  8 , 49 ,  1   )          E
   47        (  8 , 50 ,  1   )          E
 48     ( 8 , 51 ,  1  )      E
 49     ( 8 , 52 ,  1  )      E
 50     ( 8 , 53 ,  1  )      E
 51     ( 8 , 54 ,  1  )      E
 52     ( 8 , 55 ,  1  )      E
 53     ( 8 , 56 ,  1  )      E
 54     ( 8 , 57 ,  1  )      E
 55     ( 8 , 58 ,  1  )      E
 56     ( 8 , 59 ,  1  )      E
 57     ( 8 , 60 ,  1  )      E
 58     ( 8 , 61 ,  1  )      E
 59     ( 8 , 62 ,  1  )      E
 60     ( 8 , 63 ,  1  )      E
 61     ( 8 , 64 ,  1  )      E
 62     ( 8 , 65 ,  1  )      E
 63     ( 8 , 66 ,  1  )      E
 64     ( 8 , 67 ,  1  )      E
 65     ( 8 , 68 ,  1  )      E
 66     ( 8 , 69 ,  1  )      E
 67     ( 8 , 70 ,  1  )      E
 68     ( 8 , 71 ,  1  )      E
 69     ( 8 , 72 ,  1  )      E
 70     ( 8 , 73 ,  1  )      E
 71     ( 8 , 74 ,  1  )      NE
 72     ( 7 , 75 ,  1  )     DOWN
 73     ( 7 , 75 ,  2  )     DOWN
 74     ( 7 , 75 ,  3  )      SW
 75     ( 8 , 74 ,  3  )      W
 76     ( 8 , 73 ,  3  )      W
 77     ( 8 , 72 ,  3  )      W
 78     ( 8 , 71 ,  3  )      W
 79     ( 8 , 70 ,  3  )      W
 80     ( 8 , 69 ,  3  )      W
 81     ( 8 , 68 ,  3  )      W
 82     ( 8 , 67 ,  3  )      W
 83     ( 8 , 66 ,  3  )      W
 84     ( 8 , 65 ,  3  )      W
 85     ( 8 , 64 ,  3  )      W
 86     ( 8 , 63 ,  3  )      W
 87     ( 8 , 62 ,  3  )      W
 88     ( 8 , 61 ,  3  )      W
 89     ( 8 , 60 ,  3  )      W
 90     ( 8 , 59 ,  3  )      W
 91     ( 8 , 58 ,  3  )      W
 92     ( 8 , 57 ,  3  )      W
 93     ( 8 , 56 ,  3  )      W
 94     ( 8 , 55 ,  3  )      W
 95     ( 8 , 54 ,  3  )      W
 96     ( 8 , 53 ,  3  )      W
 97     ( 8 , 52 ,  3  )      W
 98     ( 8 , 51 ,  3  )      W
 99     ( 8 , 50 ,  3  )      W
100     ( 8 , 49 ,  3  )      W
101     ( 8 , 48 ,  3  )      W
102     ( 8 , 47 ,  3  )      W
103     ( 8 , 46 ,  3  )      W
104     ( 8 , 45 ,  3  )      W
105     ( 8 , 44 ,  3  )      W
106     ( 8 , 43 ,  3  )      W
107     ( 8 , 42 ,  3  )      W
108     ( 8 , 41 ,  3  )      W
109     ( 8 , 40 ,  3  )      W
```

```
110     (  8 , 39 ,  3  )     SW
111     (  9 , 38 ,  3  )     SW
112     ( 10 , 37 ,  3  )     SW
113     ( 11 , 36 ,  3  )     SW
114     ( 12 , 35 ,  3  )     SW
115     ( 13 , 34 ,  3  )     SW
116     ( 14 , 33 ,  3  )      S
117     ( 15 , 33 ,  3  )     SW
118     ( 16 , 32 ,  3  )      S
119     ( 17 , 32 ,  3  )      S
120     ( 18 , 32 ,  3  )     SW
121     ( 19 , 31 ,  3  )     SW
122     ( 20 , 30 ,  3  )      W
123     ( 20 , 29 ,  3  )      W
124     ( 20 , 28 ,  3  )      W
125     ( 20 , 27 ,  3  )      W
126     ( 20 , 26 ,  3  )      W
127     ( 20 , 25 ,  3  )      W
128     ( 20 , 24 ,  3  )      W
129     ( 20 , 23 ,  3  )      W
130     ( 20 , 22 ,  3  )      W
131     ( 20 , 21 ,  3  )      W
132     ( 20 , 20 ,  3  )      W
133     ( 20 , 19 ,  3  )      W
134     ( 20 , 18 ,  3  )      W
135     ( 20 , 17 ,  3  )      W
136     ( 20 , 16 ,  3  )      W
137     ( 20 , 15 ,  3  )      W
138     ( 20 , 14 ,  3  )      W
139     ( 20 , 13 ,  3  )      W
140     ( 20 , 12 ,  3  )      W
141     ( 20 , 11 ,  3  )     NW
142     ( 19 , 10 ,  3  )     NW
143     ( 18 ,  9 ,  3  )      W
144     ( 18 ,  8 ,  3  )      W
145     ( 18 ,  7 ,  3  )     NW
146     ( 17 ,  6 ,  3  )      W
147     ( 17 ,  5 ,  3  )      W
148     ( 17 ,  4 ,  3  )      W
149     ( 17 ,  3 ,  3  )     GOAL
```

```
Maze Solution...

  The Maze:   b3dnav.dat   has  30 Rows , 80 Cols ,  3 Depth Layer(s)

  Key:  Obstacles   = X (isolated) ,   @ (conrected)
        Free Space = .   [ + D (doors) ,   E (elevators) ]
        Start node = S , at (  3,  3,  1)
        Goal  node = G , at ( 17,  3,  3)


      12345678901234567890123456789012345678901234567890123456789012345678901234567890

Depth Layer:  1
  1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2   X........X...................X.........X...........X.................X.........X
  3   X.S......X...................X.........X...........X.................X.........X
  4   X..1.....X...................X.........X...........D.................X.........X
  5   X...2....X...................X.........X...........X.................X.........X
  6   X....34..X...................X.........X...........X.................X.........X
  7   XXXXXXX5XXXXXXXXXXXXXXXXXDXXXXXXXXDXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXXDXXXXXXX2XXXXX
  8   X.......678901234567890123456789012345678901234567890123456789012345678901.....X
  9   X..............................................................................X
 10   X..............................................................................X
 11   X....XXXXXXXXXXXXXXXXXXXXXXXXX.........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 12   X....D..........XXX...X......X.........X........X...............X....X.........X
 13   X....X..........X.....D......X.........X...XX...X..........XX...X....X.....XX..X
 14   X....XX...XXXXXXX.....X......X.........X...XX...X..........XX...X....X.....XX..X
 15   XXXXXXX...D.....X.....XXXXXXXX.........D...XX...X..........XX...X....D.....XX..X
 16   X....D....X.....X.....X......X.........X........X..........XX...X....X.........X
 17   XXX..X....X.....X.....X......X.........X........X....XXXX.......X....X.........X
 18   X....X....X.....X.....X......X.........X........X...............X....X....XX...X
 19   XXXXXXXXXXXXXXXXXXXDXXXXXXXDXX.........XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
 20   X..............................................................................X
 21   X..............................................................................X
 22   X..............................................................................X
 23   XXXXXXXDXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXXXXXXXXXXDXXXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
 24   X........X....X.....X........X.........X...........X.......XX........XXX.......X
 25   X........X....X.....X........X.........X...........X.......XX........XXX.......X
 26   X........X....X.....X........X.........X...........X.......XX........XXX.......X
 27   X..X..X..X....D.....X........X.........X..XX.......D.......XX........X.........X
 28   X..XXXX..X....X..............X...XXXX..X..XX.......X...XXXXXX......XXX....XXXX.X
 29   X........X....X..........XXX.X.........X..XX.......X...XXXXXX......XXX....XXXX.X
 30   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
Depth Layer:  2
  1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  5   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  6   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  7   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX3XXXXX
  8   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  9   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 11   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 13   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 14   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 15   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 16   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 17   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 19   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 20   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 21   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 22   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 23   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 24   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 25   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 26   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 27   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 28   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 29   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 30   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
Depth Layer:  3
  1    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2    X........X...................X.........X...........X.................X.........X
  3    X........X...................X.........X...........X.................X.........X
  4    X........X...................X.........X...........D.................X.........X
  5    X........X...................X.........X...........X.................X.........X
  6    X........X...................X.........X...........X.................X.........X
  7    XXXXXXXDXXXXXXXXXXXXXXXXXDXXXXXXXXDXXXXXXXXXDXXXXXXXXXXXXXXXXXXXXXDXXXXXXX4XXXXX
  8    X.....................................098765432109876543210987654321098765.....X
  9    X....................................1.........................................X
 10    X...................................2..........................................X
 11    XXXXXXXXXXXXXXXXXXXXXXXDXXXXXX.....3...XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 12    X....X........X..............X....4....X........X...............X....X.........X
 13    X....D........X..............X...5.....D........X...............X....D.........X
 14    X....X........X..............X..6......X........X...............X....X.........X
 15    XXXXXX........XXXXXXXXXXXXXXXX..7......XXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXX
 16    X....X........X.......X......X.8.......X........X...............X....X.........X
 17    X.G876........X.......X......X.9.......D........X...............X....D.........X
 18    X....X543.....X.......X......X.0.......X........X...............X....X.........X
 19    XXXXXXXXX2XXXXXXXXXDXXXXXXDXXX1........XXXXXXXXXXXXXXXXXXDXXXXXXX....XXXXXXXXXXX
 20    X.........10987654321098765432.................................................X
 21    X..............................................................................X
 22    X..............................................................................X
 23    XXXXXXXDXXXXXXXXXXXXXXXXXXDXXXXXXXDXXXXXXXXXXXXXDXXXXXXXXXXXXXXXDXXXXXXXXXDXXXXX
 24    X........X....X.....X........X.........X...........X.......XX........XXX.......X
 25    X........X....X.....X........X.........X...........X.......XX........XXX.......X
 26    X........X....X.....X........X.........X...........X.......XX........XXX.......X
 27    X........X....D.....X........X.........X..XX.......D.......XX........X.........X
 28    X........X....X..............X.........X..XX.......X...XXXXXX......XXX....XXXX.X
 29    X........X....X..............X.........X..XX.......X...XXXXXX......XXX....XXXX.X
 30    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```